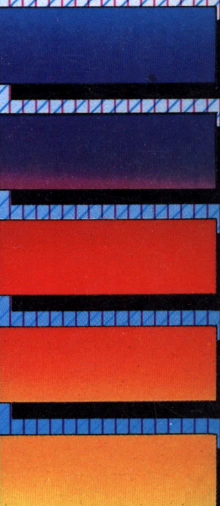


Plaza & Janés · Informática

dBASE II

Introducción al programa de tratamientos de datos

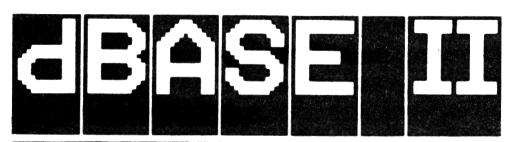
Joseph Reymann





Plaza & Janés · Informática

Joseph Reymann



Introducción al programa de tratamientos de datos

PLAZA & JANES EDITORES, S. A.

Título original: Quick and easy dBase II
Traducción de Domingo Santos

Primera edición: Octubre, 1986

Virgen de Guadalupe, 21-33
Esplugues de Llobregat (Barcelona)

Printed in Spain
Impreso en España
ISBN: 84-01-40110-0
Depósito Legal: B. 32.515 - 1986
Impreso por Industria Gráfica
Ramón Sopena, S. A.
Barcelona

1. INTRODUCCIÓN A LAS BASES DE DATOS

¿Qué es una base de datos, y qué puede hacer uno con ella?

Una *base de datos* es un grupo de ítems de información vagamente relacionados entre sí, almacenados en un ordenador. Su guía telefónica es una forma impresa de base de datos; lo mismo podemos decir del catálogo de una empresa de ventas por correo.

¿Qué *podemos* hacer con una base de datos?

Lo primero que puede hacer usted con ella es organizar, almacenar y recuperar información, *cualquier* tipo de información, desde registros comerciales a recetas de cocina, desde resultados de experimentos científicos hasta su agenda personal de teléfonos.

Y una vez tenemos esos datos en una forma organizada, ¿qué más?

Puede usted clasificarlos, buscar y encontrar los datos que desee, y analizarlos para correlaciones estadísticas entre sí. Puede hacer también cálculos con ellos: preparar sus facturas comerciales, llevar su contabilidad, escribir los cheques de los sueldos de sus empleados e imprimir sus etiquetas organizadas por códigos postales (o por el segundo apellido, o por poblaciones, o por lo que quiera).

¿Qué necesita usted para hacer todo esto? En primer lugar, necesita un ordenador. Un ordenador personal es perfectamente aceptable. En segundo lugar, necesita un programa para ese ordenador, un pro-

grama de un tipo llamado *gestor de base de datos*. El más ampliamente utilizado de esos programas es el dBASE II, un producto comercial que se halla disponible para la mayor parte de los ordenadores personales.

Este manual no pretende ser un tratamiento riguroso de la gestión de bases de datos por ordenador. Para detalles más técnicos recomendamos algún otro libro que hable de los programas de bases de datos en general. En este volumen nos centraremos en uno de los más importantes programas gestores de bases de datos para ordenadores personales: el dBASE II. De todos modos, daremos antes una breve introducción a las bases de datos en general.

El dBASE II es uno de los más útiles y poderosos programas que puede adquirir usted para su ordenador personal. En este manual lo introduciremos a algunas de las amplias posibilidades que la capacidad del dBASE II le ofrece. Empezaremos con una breve descripción de qué es una base de datos, y qué tipos de bases de datos hay. También nos tomaremos algunas libertades con el nombre; en adelante lo llamaremos simplemente dBASE (después de todo, no existe un dBASE I). En consecuencia, tenga en cuenta que cuando citemos dBASE nos estaremos refiriendo al programa específico dBASE II.

¿QUÉ ES UNA BASE DE DATOS?

Una base de datos es un conjunto de información almacenada de alguna forma y archivada en un ordenador, y a partir de la cual los datos pueden recuperarse selectivamente.

Recordemos que nuestro propósito no es simplemente tener una cierta cantidad de información. Lo más importante es disponer de *acceso* a ella, de una forma racional y eficiente. Queremos poder añadirle algunos datos y actualizar o borrar otros. El uso principal es el extraer los datos necesarios; no simplemente algunos datos, sino *todos* los datos necesarios en un

momento determinado, sin que nada quede fuera y sin tener que cargar también con datos innecesarios.

Echaremos un vistazo a los dos métodos más utilizados de estructurar datos para almacenar, gestionar y recuperar: el sistema *jerárquico* y el sistema *relacional*.

Base de datos con estructura jerárquica

Una *jerarquía* es una estructura en forma de árbol de Navidad de almacenamiento de información. Su forma habitual es parecida al organigrama de una empresa; una casilla en cada nivel forma parte de un grupo unido a una sola casilla en el nivel superior, la cual a su vez forma parte de un grupo unido a una sola casilla en el nivel superior, y así sucesivamente. Echemos un vistazo al organigrama típico de una empresa ficticia en la Figura 1.1.

Observen que cada casilla en el nivel inferior es un

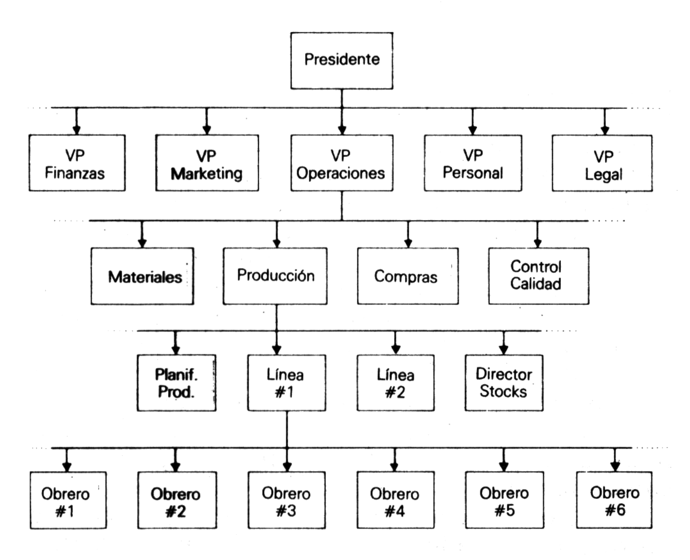


Figura 1.1. *Organigrama de una empresa ficticia.*

área de responsabilidad que no puede ser subdividida. La casilla en el nivel de arriba es descrita de una forma más general, e incluye todas las casillas de abajo con las cuales se halla unida. El otro nivel de arriba posee una descripción más general aún, y así sigue ascendiendo hasta que la descripción de la cima es tan general que incluye la responsabilidad de toda la empresa.

Esa imagen vale tanto como mil palabras si está hablando usted de organigramas comerciales y de trabajo, pero ¿cómo se relaciona esto con las bases de datos? Quizás un ejemplo concreto nos ayude. Intentemos formar una base de datos jerárquica cuya estructura incluya, como ítems de datos, todas las direcciones de las calles de los Estados Unidos. Recuerde que queremos una estructura que nos permita remover entre toda esta masa de datos y encontrar todos los datos individuales que encajen con nuestro criterio en un momento determinado. ¡Vaya trabajo! ¡Una base de datos con centenares de millones de ítems individuales! ¿Cómo podemos construir una estructura para eso?

A veces, empezar hablando de los árboles no es la mejor forma de definir un bosque. Tomemos un punto de vista más amplio. Supongamos que empezamos con una casilla en la parte superior etiquetada Estados Unidos. Debajo, podemos listar en el siguiente nivel los cincuenta estados, pero cincuenta casillas conectadas a una sola es un salto demasiado grande para un solo nivel. Supongamos que partimos los Estados Unidos en, digamos, diez regiones, cada una de ellas centrada más o menos en torno a una ciudad metropolitana importante. Ahora, dentro de esas diez regiones, podemos definir una nueva división en los estados que componen cada región. Ahora hemos bajado tres niveles (país, regiones y estados), y hemos subdividido ya nuestra base de datos en cincuenta categorías identificables de datos.

Dentro de cada estado podemos dividir de nuevo, quizá por condados, luego por ciudades. Dentro de cada ciudad podemos establecer zonas servidas por

una misma oficina postal. Desde esta oficina postal tal vez haya una docena o más de rutas individuales que, todas juntas, cubran la totalidad de las direcciones de la ciudad. Dentro de cada ruta podemos especificar las manzanas, y dentro de cada manzana las direcciones individualizadas.

Nuestra base de datos de todas las direcciones de los Estados Unidos tendrá una estructura como la de la Figura 1.2.

¿Qué hemos hecho? Tomándonos algunas libertades, hemos definido el sistema de códigos postales. Los códigos postales forman una estructura típica de base de datos jerárquica.

¿No han visto ustedes otras estructuras jerárquicas? ¿El catálogo de una gran empresa de ventas por correo, por ejemplo? Primero encontramos la mercan-

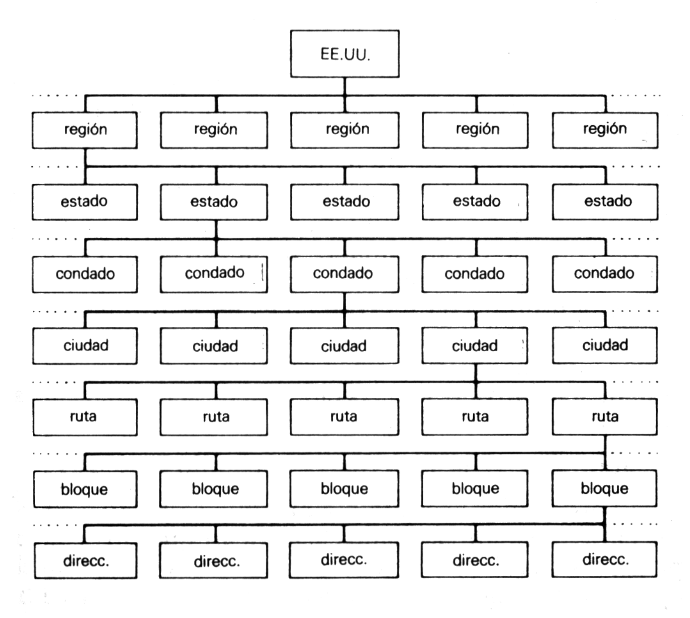


Figura 1.2. *La base de datos de todas las direcciones de calles de los Estados Unidos.*

cía pesada: muebles, grandes electrodomésticos, etc. Si buscamos el área de ferretería, encontraremos la sección de herramientas, y dentro de ésa los martillos. Vivimos en una sociedad jerarquizada, e incluso las empresas de ventas por correo siguen la estructura. (Por favor, no pase por delante de mí y mencione el índice: ¡llegaremos a ese importante atajo más tarde!)

Ahora que hablamos de estructuras, deberíamos señalar algo. ¿Para qué las tenemos? ¿No podríamos simplemente meter todos los datos dentro de una máquina y luego hacerle preguntas?

Sí, podríamos hacerlo. Supongamos que la pregunta es: «Dime quién vive en Elm Street.» Imaginemos a nuestra máquina repasando todas las direcciones de la base de datos (recuerde: centenares de millones de direcciones) y preguntándoles: «¿Es ésta una dirección de Elm Street?» ¿Cuánto tiempo tomaría eso?

Es mejor estructurar la información para fozar al usuario a especificar su pregunta con más detalles: «¿Quién vive en la Elm Street de Pocatello, Idaho?» ¿Cuánto reducimos la búsqueda? De momento eliminamos cuarenta y nueve de los cincuenta estados, y (hago una suposición) el 99,9% de las ciudades y pueblos de Idaho. El tiempo de búsqueda se reduce, quizá a una cienmilésima del que hubiera sido de la otra manera.

Así, una de las finalidades de la estructura es proporcionar alguna forma de definir qué datos significan el qué, y limitar cuánto trabajo es necesario hacer.

Otra finalidad importante es proporcionar al ordenador algunas instrucciones relativas a la interpretación de los datos. Una dirección puede ser almacenada en el ordenador de esta forma:

John Doe 1234 Elm Street Pocatello ID 83201

El ordenador necesita alguna guía respecto a cuántos caracteres almacenados significan qué: los cuatro primeros significan el nombre, los cuatro segundos el apellido, los siguientes quince son la dirección de la calle, luego nueve para la ciudad, dos para el estado, y cinco para el código postal.

Recuerde: la finalidad es *acceso*, no solo almacenamiento.

Estructura de base de datos relacional

Consideremos otro medio de almacenar información. Supongamos que almacenamos los datos como una *tabla*, en las filas y columnas que vemos tan a menudo. Si buscamos en la séptima columna de la sexta fila de una tabla en particular, encontraremos el valor 5559382. Hemos extraído un cierto dato, pero ¿qué representa eso?

He aquí alguna información general sobre tablas. Cada columna es llamada un *campo*, y tiene un nombre. A lo largo de toda la columna, los datos almacenados en ese campo son de tipo y contenido similares, aunque cada dato sea diferente y esté en distinta fila.

Cada fila es llamada un *registro*. Un registro contiene campos distintos de información, que de alguna manera se refieren a un tema en particular.

La Figura 1.3 es una representación gráfica de la tabla que hemos descrito, mostrando los campos y los registros. Observe que no ha sido llenada con ningún dato, excepto el valor que hemos encontrado en ese séptimo campo del sexto registro.

		Campos						
		F1	F2	F3	F4	F5	F6	F7
Registros	1							
	2							
	3							
	4							
	5							
	6							5559382

Figura 1.3. *Representación gráfica de una tabla.*

Volvamos ahora al dato que hemos encontrado. Veremos claro lo que es si decimos que los campos son denominados, respectivamente: apellido, nombre, calle, ciudad, estado, código del área y número de teléfono. ¿Le dice esto lo que representa el dato 5559382? Ese dato es un número de teléfono, pero ¿de quién es? Si prevalece la razón, el nombre asociado con el número telefónico es el relacionado con él por hallarse en la misma fila o registro, pero en el campo correspondiente al apellido.

Tenemos una agenda de teléfonos, una base de datos *relacional* basada en los apellidos. Cada *fichero* de datos almacena múltiples registros, cada uno de los cuales está construido de acuerdo con una *estructura* predeterminada.

La estructura es importante. Veremos en el siguiente capítulo cómo son almacenados los datos en los ordenadores. Un ordenador no posee ningún formato de tabla con filas y columnas, sólo una fila única de casilleros, *muy* larga. Tenemos que imponer una estructura de base de datos relacional en forma de una tabla con hileras y columnas en esa larga línea única de celdas de memoria.

EL DBASE II

El dBASE II, un producto de la empresa Ashton-Tate, es un poderoso programa de gestión de base de datos disponible para ordenadores personales. Es una base de datos relacional, no jerárquica. Es capaz de generar y mantener la estructura de una base de datos, y añadir información a esa base. Una vez construida, la base de datos puede ser modificada, incrementada o disminuida, clasificada y manipulada en muchas formas útiles. Contiene un metodo de encadenar juntos sus distintos comandos en *programas*, que pueden llevar por usted sus libros de contabilidad, manejar sus catálogos, generar informes, y en general proporcionarle una tremenda cantidad de ayuda. Escribir esos programas en

dBASE es similar pero mucho más sencillo que generar un programa de ordenador partiendo de cero (1).

Clasificaciones

En la gestión de bases de datos, nos referimos a menudo a *clasificar* datos. Esto significa que ese dato es situado en algún orden ascendente o descendente. Para los números, eso es fácil de comprender; la clasificación pone todos los números en orden del menor al mayor, o a la inversa. Para campos formados por caracteres de texto, clasificar significa situarlos en orden alfabético. ¿O no?

Para la representación interna de dígitos y caracteres, los ordenadores utilizan el Código Americano Estándar para el Intercambio de Información o ASCII (*American Standard Code for Information Interchange*). Una característica de este código es que las letras mayúsculas del alfabeto vienen antes que las letras minúsculas. Una M mayúscula viene antes (es decir, tiene un código inferior) de una b minúscula. Esto tiene importantes implicaciones en las clasificaciones. Si los datos contienen mezcladas mayúsculas y minúsculas, el orden de clasificación se verá alterado. MILLER vendrá antes que Marsh, por ejemplo. Esto es lo bastante importante como para referirnos de nuevo a ello, un poco más tarde.

Claves

Aquí hay que introducir un concepto adicional: la idea de una *clave* en las bases de datos.

Una *clave* no es ni más ni menos que el campo en el

(1). Recientemente ha aparecido en el mercado el dBASE III, también de Ashton-Tate. Su naturaleza es muy similar a la del dBASE II examinado aquí, si bien posee unas prestaciones muy superiores: mayor velocidad, mayor número de comandos, y grandes posibilidades en el manejo simultáneo de ficheros. *(N. del E.)*

que está usted específicamente interesado. Por ejemplo, antes formulamos la pregunta: «¿Quién vive en Elm Street?» Si buscamos en la base de datos, el elemento más importante de información que deseamos es obviamente el nombre de la calle; el campo *dirección* es la clave sobre la que se realiza la búsqueda. Si en cambio deseamos saber cuántos Jones hay, entonces nuestra clave será el campo llamado *apellido*.

Observe que puede haber más de una clave. En el directorio telefónico, los nombres están clasificados no sólo por el apellido, sino también por el nombre de pila e incluso por la inicial intermedia. Obviamente, los apellidos aparecen en orden alfabético de Able a Zilch, pero dentro de un grupo de apellidos idénticos, como Jones, queremos los nombres de pila clasificados de Alan a Zachary. Observe que cuando tenemos más de una clave, esas claves deben tener un orden de precedencia. Así, en nuestro ejemplo, la *clave primaria* será el campo «apellido», y «nombre de pila» será la clave secundaria. Si es utilizada, la inicial intermedia será una *clave terciaria*. El gestor de la base de datos clasificará, por ejemplo, según la clave primaria. Si hay conflicto con dos datos diferentes poseedores de la misma información, entonces el conflicto es resuelto comparando la clave secundaria, y así sucesivamente (ver Figura 1.4.)

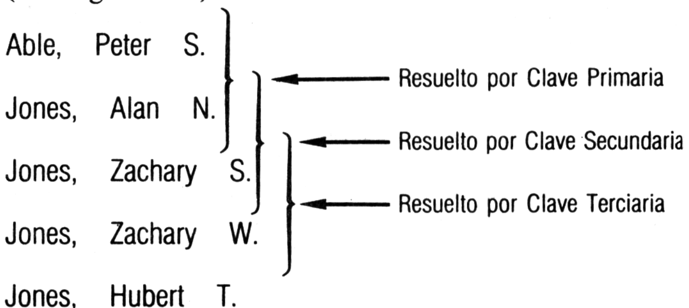


Figura 1.4.

2. CÓMO MANEJAN LOS ORDENADORES LAS BASES DE DATOS

Si ha utilizado usted alguna vez una base de datos manual (¿y quién no?), sabe cómo son actualizadas a mano. Usemos como ejemplo el sistema de catálogo de obras de una biblioteca.

Cuando es añadido un nuevo libro a la colección, como mínimo se elaboran tres fichas: una ficha de título, una ficha de autor, y al menos una ficha de tema. Esas fichas son archivadas en las secciones respectivas del catálogo de la biblioteca. El indexado de esta información se efectúa normalmente por orden alfabético en cada sección.

Para encontrar lo que usted busca en este sistema, necesita al menos una de las tres piezas de información: el título del libro que desea, o el autor, o el tema de que trata. Si mira en la sección correspondiente, es probable que encuentre lo que busca.

La asequibilidad de los ordenadores para el almacenamiento y recuperación de la información abrió nuevas posibilidades, que resultaban difíciles o imposibles de conseguir utilizando técnicas manuales. Afortunadamente, los autores de dBASE hicieron más que simplemente mecanizar el sistema manual; se alejaron de los sistemas existentes y lo enfocaron de una manera completamente distinta, y luego escribieron un programa para ordenador con mucha mayor utilidad que cualquier otro elaborado previamente.

A fin de que pueda apreciar lo que el dBASE hace

por usted, empleemos unos cuantos párrafos en describir las capacidades y limitaciones del equipo en el que el dBASE realiza su magia. Nuestra exposición del hardware del ordenador será necesariamente breve, por supuesto, ya que ésta no es la finalidad del manual que tiene entre sus manos.

QUÉ HAY DENTRO DE UN ORDENADOR PERSONAL

Un ordenador pequeño debe tener las mismas cinco partes esenciales de cualquier ordenador:

1. Una unidad central de proceso (CPU), el cerebro donde se efectúa realmente todo el trabajo de un ordenador.

2. Una memoria, la serie de *casillas de almacenamiento* temporal numeradas donde almacena la CPU sus datos.

3. Alguna forma de almacenamiento permanente para datos y programas, normalmente un sistema de almacenamiento en discos.

4. Un *programa*, una secuencia de increíblemente detalladas instrucciones que le dicen a la CPU cómo efectuar un trabajo determinado.

5. Algunos medios que permitan al operador comunicarse con y controlar al ordenador, normalmente un teclado y una pantalla.

La CPU es el corazón de todo ordenador. Es allí donde se efectúa el trabajo y son tomadas las decisiones. Algunas CPU son mejores que otras: una CPU de 16 bits puede ocuparse del doble de información a la vez que una CPU de 8 bits, lo cual constituye una ventaja en algunas tareas (pero no en todas). En general, una

CPU que actúe a mayor velocidad, como a 5 MHz (¡lo cual significa cinco millones de pulsaciones por segundo!) pude hacer más trabajo que una que funcione a un ritmo de 2 ó 4 MHz.

La cantidad de memoria interna del ordenador, a la que podemos referirnos como RAM (de *random-access memory*, memoria de acceso aleatorio), determina cuántos datos puede examinar la CPU al mismo tiempo. La RAM es muy similar a un grupo de casillas numeradas alineadas en una sola fila, donde el número secuencial de dichas casillas o celdas es llamado su *dirección*. Cada casilla de memoria almacena un carácter de información, y el dBASE de su ordenador puede disponer de 32.000 a medio millón de tales casillas, o más. Cuanta más memoria mejor, porque la CPU tan sólo puede tratar directamente con la información a la que tiene acceso en esas casillas RAM. El problema es que esas casillas olvidan su contenido cuando el ordenador es desconectado. Por ello es necesario algún método de almacenamiento a más largo plazo.

Esa necesidad es resuelta por el sistema de almacenamiento permanente del ordenador. Los primeros ordenadores personales (e incluso algunos de los más recientes) utilizan cintas a cassette. Un sistema mejor, más rápido pero también más caro utiliza discos magnéticos llamados *diskettes* para almacenar datos y programas. La CPU controla esos discos del mismo modo que controla el resto del ordenador, pero la CPU no puede manipular caracteres individuales almacenados en los discos. El grupo más pequeño de datos de caracteres que puede manejar la CPU hacia o del disco es, según sea el ordenador, de 128 a 512 caracteres en un grupo llamado *bloque*. La CPU toma uno o más bloques de datos del disco, realiza el trabajo que tenga que hacer con ellos, y luego los devuelve al disco siguiendo las instrucciones que le dé el programa.

El dBASE, por supuesto, es el programa que el ordenador ejecuta para generar y manejar datos. Usted utilizará el dBASE para crear una estructura de base de datos o señalar lo que hay almacenado en un disco,

y luego añadirá información a esa base de datos en la forma de *registros*, cada uno de los cuales contendrá toda la información relativa a un tema en particular. Esos registros serán almacenados también en el disco.

Lo bien y lo rápido que actúe el dBASE es determinado en buena parte por las capacidades del ordenador en que es ejecutado el programa. En particular, uno de los factores más importantes es la velocidad en que puede escribir y leer información del sistema del disco. Puesto que el almacenamiento interno de la memoria de la mayor parte de ordenadores es muy limitada comparada con el almacenamiento externo, en la mayor parte de los casos el dBASE deberá trabajar enormemente con los discos. Tareas tan necesarias y frecuentes como seleccionar y buscar pueden tomar mucho tiempo, o ser muy rápidas, según la velocidad del sistema de discos en su ordenador.

ALMACENAMIENTO DE DATOS EN EL ORDENADOR

En el siguiente capítulo entraremos en detalles de construir la estructura de una base de datos. Por ahora, simplemente queremos decirle lo que hace el ordenador con la información que almacena.

Hay varios métodos utilizados por los especialistas y programadores para almacenar datos que deben ser recuperados más tarde. Cada uno de ellos tiene sus ventajas y sus inconvenientes. El dBASE emplea uno de los métodos más simples, que utiliza muy poca memoria *de estructura*: el espacio reservado no a almacenar datos sino a almacenar los medios para localizar esos datos.

El dBASE controla una amplia sección de la memoria interna de su ordenador. Al principio de ese espacio, almacena la *estructura de registros*: la información que le proporciona usted al definir un registro individualizado de esa base de datos en particular. El resto del espacio es utilizado para almacenar los datos de los registros. El largo bloque de casillas de memo-

ria no está organizado de ninguna manera en particular. Los datos son almacenados secuencialmente como un largo grupo de caracteres, uno tras otro. ¿Cómo sabe el dBASE dónde empieza un registro y termina otro? ¿Y cómo sabe el dBASE dónde, dentro de un registro en particular, se halla almacenado un determinado dato?

El dBASE sabe dónde está almacenado el primer carácter de la base de datos, y ése es el principio del primer registro: la *dirección inicial*. Digamos para un ordenador en particular que la dirección inicial es la localización 10000. La estructura definida por usted para un registro de datos le dice al dBASE el número de caracteres de cada campo. En consecuencia, las direcciones donde empieza cada campo en particular pueden hallarse tomando la dirección inicial de ese registro y añadiéndole las longitudes de cada uno de los campos que anteceden al que deseamos. Si el primer campo de cada registro es el nombre de pila de una persona, y para ese nombre,han sido reservados diez caracteres, entonces el segundo campo, el apellido, empieza en 10000 (la dirección inicial del registro) más 10 (la longitud del primer campo), o sea 10010.

La localización dentro de la memoria del ordenador de cada campo de este primer registro puede ser calculada de forma similar, a partir de la estructura que define la longitud de cada campo.

La estructura define también la longitud de un registro. Esta longitud es un carácter más que la suma de las longitudes de todos los campos de ese registro, puesto que el dBASE inserta un carácter de delimitación entre los registros. Si ha definido usted una base de datos en particular como compuesta por registros de 116 caracteres en total (incluido el carácter entre registros) para cada uno de ellos, y el primer registro empieza en la dirección 10000, entonces el segundo registro empezará en 10116, que es la dirección inicial más la longitud del primer registro. Dentro de este segundo registro, el campo del apellido empieza en 10116 más 10, o sea 10126.

Podemos ver pues que el dBASE, sabiendo tan sólo la dirección inicial del área de memoria reservada al almacenamiento de datos y las longitudes de un registro y de cada uno de sus campos, puede localizar cualquier campo dentro de cualquier registro mediante una serie de sumas. Si desea usted visualizar esto, supongamos que tiene una cuerda de cincuenta pies de largo, bien tensa, y una regla de un pie subdividida en pulgadas. ¿Cómo localizar la pulgada 278 de esa cuerda?

Una forma es señalar longitudes de un pie con la regla, sumando doce pulgadas, que es lo que tiene un pie, cada vez que lo haga. Cuando haya recorrido en la cuerda veintitrés pies, sabrá que ha alcanzado la pulgada 276. En consecuencia, la pulgada 278 es la segunda después del inicio del pie 24.

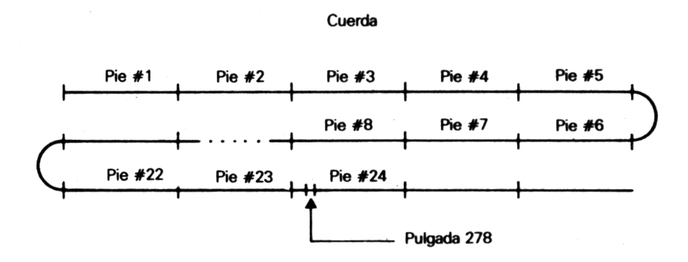


Figura 2.1. *Localizando la pulgada 278 de una cuerda.*

El dBASE hace lo mismo para localizar los datos almacenados, como sobre una cuerda, a lo largo de la hilera de caracteres de la memoria. Utiliza la estructura de registros definida por usted como el equivalente de la regla sobre la cuerda, situándola a lo largo de la hilera de datos de caracteres almacenados en sucesivas localizaciones de la memoria. Moviendo toda la estructura de esa «regla» a lo largo de la «cuerda», puede seleccionar los distintos registros. Buscando en una «pulgada» o campo específico dentro de esa «regla», puede encontrar cualquier ítem determinado de información.

Compruebe que ha comprendido este método de localización calculando dónde localizar el campo del apellido del registro dieciséis en esa estructura de registros de 116 caracteres. Deberá obtener como resultado la dirección 11750.

3. CONSTRUYENDO UNA BASE DE DATOS dBASE II

Para poder hablar en términos concretos, vamos a utilizar ejemplos prácticos a lo largo de todo este manual. En este capítulo crearemos una base de datos con los registros del personal de un pequeño negocio propiedad de Roger y Emma Ransom. Haciendo esto, examinaremos los comandos dBASE CREATE, APPEND, COPY, EDIT, MODIFY, INSERT, BROWSE, DELETE y PACK.

En capítulos sucesivos escribiremos un programa simplificado utilizando los comandos dBASE para preparar la nómina semanal de esa base de datos.

El dBASE es un programa de ordenador del tipo llamado *gestor de base de datos*: maneja y gestiona datos. Para efectuar este trabajo, tiene que organizar los datos en una *estructura*, de modo que el programa pueda interpretar correctamente la información almacenada en el ordenador y dada al ordenador por el operador para que la almacene. En consecuencia, primero deberemos definir su estructura.

DISEÑANDO UNA BASE DE DATOS

Antes hemos descrito brevemente cómo almacena los datos un ordenador. El dBASE conserva sus datos como una larga hilera de caracteres. Cuando necesita intepretar o utilizar los datos, quizá para preparar

informes o presentarle a usted una pantalla de información, el programa revisa esta larga hilera de datos mediante una «regla» o plantilla a través de la cual puede ser interpretada correctamente la hilera.

El primer paso para utilizar el dBASE consiste en definirle al ordenador cuál va a ser esa plantilla. A eso se le llama «crear» una base de datos, y de hecho éste es el comando que utilizaremos: CREATE. Una vez introducido correctamente el dBASE en nuestro ordenador, la máquina nos saludará con un mensaje que contiene el sello de la marca dBASE y el símbolo del punto (.). Este símbolo del punto es utilizado en el dBASE como un *prompt*, una señal que le indica al operador que el programa está aguardando nuevas instrucciones. Nuestra intención es crear una estructura de base de datos, así que teclearemos:

```
CREATE PERSONAL
```

Esto le dice al dBASE que queremos definir una nueva base de datos llamada PERSONAL.DBF. Tenga en cuenta que los nombres de los ficheros están limitados a ocho caracteres en los sistemas operativos de muchos ordenadores personales, por lo que si el nombre es más largo deberemos abreviarlo. Así, CORRESPONDENCIA tendrá que ser CORRESPD o CORRSPND o cualquier otra abreviatura que usted elija y que no supere los ocho caracteres. Aunque usted no especifique la extensión .DBF en el comando CREATE, el dBASE utiliza .DBF como extensión implícita de *Data Base File*, Fichero de Base de Datos, siempre que usted no especifique algún otro tipo distinto de fichero. (Vea en esta misma colección el volumen *CP/M* para un examen más detallado de los nombres y tipos de ficheros.)

El dBASE quiere ahora que le definamos una plantilla para cada campo de un registro típico de PERSONAL. Recuerde que, aunque el contenido de los registros individuales es diferente, la estructura y longitud de todos los registros de una base de datos en particu

lar son siempre las mismas. Si definimos los campos de un registro, estamos definiendo los campos de todos los registros de esa base de datos.

Definir los campos requiere que le demos al dBASE un nombre para cada campo, una indicación de si el contenido será caracteres de texto, números o indicadores lógicos (verdadero o falso), y una cifra del número máximo de caracteres en ese campo. Para los campos numéricos, puede especificarse también el número de dígitos que el dBASE debe suponer que se hallan a la derecha de la coma decimal (cero si no se especifica algo distinto).

En respuesta a su comando de crear, el dBASE nos dice:

```
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD    NAME,TYPE,WIDTH,DECIMAL PLACES
001
```

o sea: «Entre estructura del registro como sigue: campo nombre,tipo,anchura,lugares decimales 001», y aguarda a que empecemos a definir la estructura. Podemos disponer de hasta 32 campos en cada registro, pero no necesitamos tantos para nuestro ejemplo de base de datos y programa. Teclearemos la información que necesita el dBASE, empezando con los nombres de los campos hasta un total de diez caracteres para cada uno. Eligiremos nombres que sean descriptivos de su contenido, para ayudarnos más tarde a recordar los datos y programar los informes y cálculos. Puesto que no se permiten espacios dentro de un nombre, una práctica típica es utilizar los dos puntos (:) como separador entre nombres de dos palabras. El tipo de información a almacenar es señalada por un indicador de una sola letra: C para carácter, N para número y L para lógico.

Para nuestro ejemplo, estructuraremos la base de datos de la siguiente manera:

```
 FIELD   NAME,TYPE,WIDTH,DECIMAL
PLACES
  001    NUM:EMPL,C,4
  002    ANTIGDAD,C,6
  003    EXTN:TEL,C,4
  004    NOM:PILA,C,10
  005    APELLIDO,C,15
  006    CALLE,C,20
  007    CIUDAD,C,12
  008    ESTADO,C,2
  009    COD:POSTAL,C,5
  010    TEL:PART,C,11
  011    NO:SEG:SOC,C,9
  012    EXENC,N,2
  013    FCHA:NAC,C,4
  014    SUELDO,N,5,3
  015    HRAS:SEM,N,3,1
  016    BRUT:SEM,N,6,2
  017    BRUT:ANL,N,7,2
  018    IMPFED:SEM,N,6,2
  019    IMPFED:ANL,N,7,2
  020    IMPEST:SEM,N,5,2
  021    IMPEST:ANL,N,6,2
  022    SEG:ANL,N,6,2
  023    NET:SEM,N,7,2
  024    NET:ANL,N,7,2
  025    FECH:FIN,C,6
  026
```

Cuando alcancemos el *prompt* para el campo #2(pulsaremos simplemente la tecla RETURN o ENTE] para decirle al dBASE que hemos terminado con l definición. El dBASE nos preguntará:

```
INPUT DATA NOW?
```

queriendo saber si vamos a entrar inmediatamen datos. Por el momento pulsaremos N (es decir, NO).

Lo que hemos hecho hasta ahora ha sido crear la estructura para una base de datos. Todavía no existe ninguna base de datos propiamente dicha, solamente la plantilla de una. Observe algunas cosas de interés en nuestra estructura. Número de empleado, antigüedad, extensión telefónica y código postal son todos ellos números. Sin embargo, probablemente no vamos a efectuar ningún cálculo con esos campos, así que los hemos identificado como campos de caracteres. De todos modos, podremos clasificar los datos de esos campos de modo que nos den, por ejemplo, una lista de empleados ordenados por antigüedad en el empleo. Hemos almacenado también el nombre de los empleados en dos campos separados, nombre y apellido. Esto nos permite elaborar listados como una relación de extensiones telefónicas ordenadas alfabéticamente por apellidos. Del mismo modo, separando ciudad y estado del resto de la dirección, podremos clasificar al personal por ciudades para tener información a la hora de establecer un itinerario para el autobús del transporte. Por supuesto, para negocios pequeños no localizados cerca de la frontera interestatal, el campo ESTADO puede eliminarse.

El campo SUELDO tiene tres decimales porque el salario para algunas horas está fijado en 4,685$.

Un modo útil de especificar las fechas es en la forma AAMMDD, donde AA son dos dígitos que indican el año, MM otros dos dígitos que idican el mes (1-12), y DD los dos dígitos del día. Así, 840116 señalará el 16 de enero de 1984. Las fechas especificadas de esta forma son más fáciles de clasificar. Observe que la información de la fecha de nacimiento almacena solamente el mes y el día (MMDD), no el año de nacimiento, para evitar controversias y malas interpretaciones sobre discriminación por motivos de edad.

Esta estructura de base de datos contiene en el registro de cada empleado toda la información clave necesaria: nombre y dirección, información interna como la extensión telefónica y el número de empleado, e información económica como los impuestos y exen-

ciones sobre el suelo, proporcionando también campos para acumular datos anuales. La finalidad general al definir una base de datos es hacer esos datos tan útiles como sea posible para una amplia variedad de propósitos.

TRABAJANDO CON LA BASE DE DATOS

ENTRANDO DATOS EN LA BASE

Ahora que hemos definido la estructura, podemos empezar a entrar información. Hay dos formas de hacerlo: entrando registros individuales, o traspasando los datos de otra base de datos.

El dBASE contiene un comando APPEND, que permite añadir registros a la base de dados. Primero, tiene que decirle al dBASE qué base debe utilizar tecleando:

```
USE PERSONAL
```

El dBASE abrirá ese fichero. Luego, si teclea:

```
APPEND
```

le mostrará el número del siguiente registro (1, en una base de datos vacía) y la estructura del registro. El dBASE aguardará ahora a que usted le entre datos.

No necesita llenar todos los espacios disponibles en un campo, como tampoco tiene que entrar algo en cada campo. Si el nombre de pila de un empleado es Sam, simplemente entre esas tres letras y pulse RETURN o ENTER. El dBASE le situará automáticamente en el siguiente campo; sin embargo, el dBASE almacenará todos los diez caracteres del campo, llenando el resto con blancos.

Puede entrar usted tantos datos como tenga. Si no tiene los datos de un campo en particular, sáltese ese

campo y éntrelo más tarde, cuando disponga de ellos. Para el sistema puesto como ejemplo en este manual, hemos utilizado los datos ficticios que aparecen en el ejemplo de base de datos que figura en el apéndice, en las páginas 109-118.

Al entrar los datos, es importante establecer un procedimiento y un estilo y atenerse a él. Puede utilizar letras en mayúsculas o en minúsculas. Aunque es posible que más tarde desee que todos los nombres figuren con mayúsculas (para imprimir los cheques de los salarios, por ejemplo), puede entrar el nombre con sólo la inicial en mayúsculas. Pueden darse fácilmente instrucciones al dBASE para que en un momento dado ponga todas las palabras en mayúsculas. Sin embargo, entre los datos en los correspondientes campos de cada registro de la misma manera (todo en mayúsculas, o con las iniciales en mayúsculas, o como quiera) a lo largo de toda la base de datos. No importa si el estilo es distinto para distintos campos, siempre que en cada registro cada campo en particular tenga el mismo estilo a lo largo de toda la base de datos. Cuando lleguemos al comando SORT (clasificar), explicaremos por qué.

Cuando haya entrado el último de sus datos, termine la sesión de añadir pulsando un RETURN o ENTER al principio del *prompt* de un nuevo registro. Asegúrese de que cierra el fichero tras cada utilización. Del mismo modo que ha tecleado USE con el nombre del fichero para abrirlo, teclee USE (sin el nombre del fichero) para cerrarlo.

INSERTANDO DATOS

APPEND añade siempre los datos al final de la base de datos. Pero pueden presentarse ocasiones en que desee usted añadir un registro en mitad de la base de datos. Para ello, el dBASE ofrece el comando INSERT.

En el siguiente capítulo veremos cómo posicionarse en diferentes registros dentro de la base de datos.

Por ahora, simplemente supondremos que se ha posicionado usted en un registro determinado. Podemos INSERTar otro registro antes o después de ese registro con sólo teclear INSERT o INSERT BEFORE. Al igual que con el comando APPEND, el dBASE le dará un *prompt* pidiéndole los datos para ese registro y mostrando la estructura del registro. Si por alguna razón desea usted insertar un registro en blanco (uno sin datos), teclee INSERT BLANK, y el dBASE completará la acción sin mostrar ningún *prompt*.

Añadiendo datos de otra base de datos

También es posible añadir datos de una base de datos ya existente a su base de datos PERSONAL. Si dispone usted de otro fichero de datos con una estructura similar aunque no necesariamente idéntica, puede mezclar los datos de ese fichero en sus registros PERSONAL. Simplemente teclee:

APPEND FROM <nombrefichero>

y los datos del *nombrefichero* serán añadidos a PERSONAL. ¿Pero y si la estructura del otro fichero es distinta? Solamente aquellos campos que posean nombres idénticos en ambas bases de datos serán trasladados. Si ambas bases de datos poseen un campo NOM:PILA, serán creados nuevos registros con la información de esos nombres de pila. Cada campo con el mismo nombre en ambas bases será trasladado a nuevos registros en el fichero PERSONAL.

Modificando una estructura existente de base de datos

Esta técnica es también una de las formas de amplia o modificar bases de datos existentes. Cuando hube crea do los datos ficticios para la base de datos del ejemplo d este manual, me di cuenta de que añadir el campo FCH

NAC podía generar fácilmente una listado clasificado por fechas de cumpleaños que el director de la compañía podría utilizar para enviar la correspondiente felicitación a sus empleados. Pero ya había generado la estructura de la base de datos sin FCHA:NAC, y terminé de teclear todos los datos ficticios. Si la estructura de una base de datos es cambiada, todos los datos existentes estarán almacenados en el ordenador de acuerdo con la antigua estructura, pero cualquier petición de datos se esperará que tenga en cuenta la estructura nueva. En consecuencia, la salida de datos se verá embrollada. ¿Hay alguna forma de salirse de esto? Sí, la hay.

Puesto que modificar la estructura de una base de datos ya estructurada embrollará sin lugar a dudas todos los datos que contenga, el dBASE obvia el problema simplemente eliminando todos esos datos. Yo deseaba no perder mi antigua base de datos, a fin de no tener que teclearlo todo de nuevo. Para salvar los datos existentes, generé una nueva base, TEMPO, tecleando estos comandos:

```
USE PERSONAL
COPY TO TEMPO
```

Esto crea la base de datos TEMPO, con una estructura idéntica a PERSONAL y con todos los datos de PERSONAL. Luego el dBASE me permite modificar la estructura de PERSONAL si tecleo:

```
USE PERSONAL
MODIFY STRUCTURE
```

Para advertirle y darle una segunda oportunidad de detenerse si ha tecleado usted algún comando erróneo, el dBASE se asegura de que lo que quiere usted es desechar todos los datos existentes preguntando:

```
MODIFY ERASES ALL DATA RECORDS...;
  PROCEED? (Y/N)
```

Puesto que yo ya había salvado una copia de los datos en el fichero TEMPO, no necesitaba salvar los datos en PERSONAL. Pulsé Y, y el dBASE eliminó todos los datos de PERSONAL, presentando un display de la estructura existente. Hice mi cambio, insertando el campo FCHA:NAC y tecleando W (control W) para almacenarlo.

Ahora, con la NUEVA estructura ya en el fichero PERSONAL, llamé todos los datos almacenados temporalmente en TEMPO al nuevo fichero PERSONAL tecleando:

```
USE PERSONAL
APPEND FROM TEMPO
```

Toda la base de datos original fue trasladada de su almacenamiento temporal en el fichero TEMPO y almacenada correctamente en PERSONAL bajo la nueva estructura. Lo que tenía ahora era el fichero PERSONAL con todos los datos antiguos, pero conteniendo el campo FCHA:NAC, que estaba vacío de datos. Era una tarea sencilla añadir la información de las fechas de nacimiento a cada registro utilizando el procedimiento EDIT descrito más abajo.

Supongamos que cuando modifiqué la estructura de PERSONAL añadí el campo FCHA:NAC, pero al mismo tiempo borré el campo ANTIGDAD. Al copiar los datos con el comando APPEND, los datos en ese campo de TEMPO serían simplemente pasados por alto por el dBASE.

Las poderosas características del dBASE permiten sin la menor duda construir y modificar fácilmente una base de datos.

MODIFICANDO LA INFORMACIÓN DE SU BASE DE DATOS

Finalmente tendrá usted una base de datos llena de información, y descubrirá que necesita cambiar algo. En su fichero PERSONAL, por ejemplo, puede espera-

cambios periódicos en el campo SUELDO a medida que los empleados asciendan de categoría. Si más tarde desea añadir datos como la información de la fecha de nacimiento a registros ya existentes o cambiar datos que ya existen allí, puede EDITar registros individuales. El dBASE lleva incorporado un editor, que le permite utilizar las teclas de control de su ordenador para moverse por la pantalla, cambiando datos en un registro. La tabla 3.1 muestra las principales teclas de control y sus acciones. (Observe que una tecla de control como la control-W está indicada: ^W. Esto significa mantener apretada la tecla **Ctrl** de su teclado al tiempo que pulsa la letra correspondiente.)

Tras abrir el fichero con USE PERSONAL, teclee EDIT y el número del registro que desea cambiar. Ese registro le será mostrado en la pantalla para ser cambiado. ¿Cómo encontrar el número del registro? Uti-

Tabla 3.1. *Teclas EDIT del dBASE II*

A	Retrocede un campo (flecha arriba en el IBM-PC)
C	Pasa al siguiente registro (PgDn en el IBM-PC)
^D	Avanza un espacio (flecha a la derecha en el IBM-PC)
^E	Retrocede un campo (flecha arriba en el IBM-PC)
^F	Avanza al siguiente campo (flecha abajo en el IBM-PC)
^G	Borra el carácter bajo el cursor
^N	Inserta un nuevo campo, desplazando los otros
^P	Conecta y desconecta la impresora del ordenador
^Q	Final: aborta el trabajo, *desechando* todos los nuevos datos
^R	Se traslada al registro anterior (PgUp en el IBM-PC)
^S	Retrocede un espacio (flecha a la izquierda en el IBM-PC)
^T	Borra el campo
^U	Señala este registro para ser borrado (es un *toggle*: ^U de nuevo restablece el registro)
^V	Pasa de modo reescribir a modo insertar y viceversa
^W	Final: abandona el trabajo, salvando todos los nuevos datos (^O en algunas versiones del dBASE)
^X	Avanza un campo (flecha abajo en el IBM-PC)
^ltr	significa pulsar la letra dada mientras se mantiene apretada la tecla de Control.

Si su ordenador posee una tecla «RUB», borrará el carácter a la izquierda del cursor.

lice el comando BROWSE (vea más abajo) para encontrarlo.

Mientras está editando, puede usted avanzar o retroceder un registro completo pulsando ^C o ^R. Cuando haya terminado, pulse ^W para salvar sus datos. Vaya con cuidado: ¡la tecla Q está cerca de la tecla W en su teclado! ¡Pulsar ^Q por error eliminará todos los cambios que haya editado!

Hay otra forma de cambiar datos dentro de la base de datos. El comando EDIT es utilizable solamente cuando se halla usted en modo interactivo controlado por el operador. No puede ser usado desde dentro de un programa. El comando REPLACE del dBASE puede cambiar datos en registros mientras se está ejecutando un programa. Este comando puede operar o bien sobre el registro que haya sido seleccionado por el programa, o sobre una serie de registros especificados anticipadamente por el programador. Veremos un ejemplo utilizando REPLACE en nuestro programa de nómina en el capítulo 6.

BROWSE (HOJEE) SUS DATOS

El dBASE posee con comando muy poderoso que le muestra sus datos y le permite editarlos. La palabra que lo define es explícita: BROWSE, hojear. Una vez ha seleccionado usted su base de datos con el comando USE, el comando BROWSE le mostrará una pantalla llena de datos de una sola vez, permitiéndole revisarla por entre campos y registros, y cambiar el contenido de los campos que desee utilizando los comandos de edición del dBASE. Si un registro tiene más de ochenta caracteres (la anchura de la mayor parte de las pantallas), solamente serán mostrados los ochenta primeros caracteres. ^B y ^Z pueden ser usados entonces para mover su «ventana» a la derecha o a la izquierda respectivamente y mostrar otras zonas del registro. Una vez efectuados sus cambios, teclee ^W para salvarlos, o ^Q si desea eliminarlos sin salvarlos.

DESECHANDO DATOS VIEJOS

Finalmente, llegará un momento en el que probablemente algunos datos de su base de datos deberán ser desechados; pueden haber quedado obsoletos, o simplemente ser irrelevantes para la finalidad para la que es usada la base de datos ahora. En nuestro ejemplo, solamente deseamos mantener registros de los empleados en activo.

Antes de ver cómo eliminar registros, queremos mencionar la necesidad de retener muchas veces esos registros, especialmente en los negocios. A veces, con los registros de personal auténticos, puede que desee usted guardar la información aunque la persona ya no esté empleada en su empresa. No entraremos en detalles en este aspecto, puesto que se halla más allá del alcance de este libro.

El dBASE proporciona un par de formas de manejar los borrados. Si sabe usted el número del registro que desea eliminar, el simple comando es DELETE RECORD *n*. Si no sabe el número, primero tiene que encontrar el registro. Si, y solamente si, está seguro de que puede localizar el registro que le interesa mediante una expresión, puede teclear algo así como:

```
DELETE FOR NOM:PILA = "JUAN" .AND.;
  APELLIDO = "OCHOA"
```

Sin embargo, de este modo pueden producirse errores. Un enfoque mejor es utilizar el comando DISPLAY con la misma expresión de arriba para mostrar primero el dato, con el número de registro, y confirmar así que se trata realmente del registro que se quiere borrar. Luego DELETE por el número de registro.

El dBASE no extirpa en realidad el dato de su fichero durante una operación DELETE. El registro en particular queda simplemente *señalado* para ser borrado, y muchos comandos dBASE como el APPEND y el SORT ya no reconocerán su existencia. El dBASE contiene una *banderola*, un indicador que le permite a

usted seleccionar si algunos comandos dBASE reconocerán o no un registro borrado con el DELETE. Teclear SET DELETED ON informará al dBASE que no desea que esos registros sean tenidos en cuenta por FIND, LOCATE, LIST, COUNT y comandos parecidos. Teclear SET DELETED OFF (que es la condición implícita del dBASE si usted no especifica nada) permite localizarlos por algunos comandos. Así, puede usted llevar a pantalla un registro borrado, que aparecerá con un asterisco después del número del registro.

Puesto que los datos están todavía ahí, esos regis-tros pueden ser recuperados si han sido borrados inco-rrectamente. Puede dar usted instrucciones al dBASE de que RECALL RECORD 4, por ejemplo, para recupe-rar el registro 4, o bien RECALL FOR APELLIDO = "OCHOA". Incluso puede tomar el camino más corto y pedir RECALL ALL, es decir, todo.

El comando que extirpa realmente los registros bo-rrados es PACK. Una vez completada esta operación los registros borrados están más allá de toda recupe-ración. La operación PACK se efectúa sobre el ficher existente, de modo que no queda ninguna copia. U procedimiento alternativo al PACK es COPY, copiar e fichero después del borrado. Por ejemplo, tras borra algunos registros, puede usted COPY PERSONAL T PERSONAL2. Con este método, el viejo fichero PEI SONAL sigue existiendo, con la información borrad Puesto que COPY no reconoce los registros borrados, nuevo fichero PERSONAL2 no señalará ningún regi tro como marcado para ser borrado.

Si elige usted este método, puede que desee (o te ga que) renombrar los ficheros. Por ejemplo, puede t ner usted un programa de nómina que espera que información de los empleados en activo se halle sie pre en un fichero llamado PERSONAL.DBF. Podría s prudente renombrar el fichero antiguo con algo a como ANTIGPERS.DBF, y luego renombrar PERS NAL2.DBF como PERSONAL.DBF.

Observe que, en el teclado, las teclas de control c cursor E, S, D y X forman un rombo, con las puntas

ñalando la dirección del movimiento que producirá el pulsar esa tecla.

A efectos de compatibilidad con la filosofía de control de programas de proceso de texto tales como el WordStar, las teclas ^A y ^F tienen el mismo efecto que ^E y ^X respectivamente: retroceden o avanzan una «palabra» o campo.

4. RECUPERANDO DATOS

En el capítulo 2 vimos cómo almacenan los datos los ordenadores. Entre el ordenador en sí y los discos magnéticos hay entre 100.000 y dos millones de casillas almacenando esos datos. Este capítulo introducirá conceptos tales como la forma en que el dBASE efectúa su principal tarea: manejar todo ese espacio de almacenamiento de modo que usted pueda acceder en cualquier momento al dato que desea. El dBASE consigue esto mediante *comandos*, que puede usted teclear o almacenar en la forma de un *programa dBASE*. En este capítulo introduciremos algunos de los comandos básicos disponibles. Veremos los comandos USE, CLEAR, SELECT, DISPLAY, LIST, GOTO, SKIP, STORE, TRIM, SORT, INDEX, LOCATE y FIND. Introduciremos también el comando SET, una palabra con muchos usos y significados distintos en el dBASE. En el capítulo 6 examinaremos cómo acumular esos y otros comandos en programas.

USEando SUS DATOS

Lo primero que hay que hacer cuando desee usted manejar de alguna forma los datos que ha almacenado es informar al dBASE de con qué fichero desea trabajar. Lo hará diciéndole al dBASE que USE ese fichero. Para nuestro ejemplo, teclee USE PERSONAL. El

dBASE hará entonces varias cosas. En primer lugar, abrirá ese fichero, y luego se trasladará a su primer registro. A partir de entonces el dBASE utilizará ese primer registro como referencia para todas las futuras operaciones con datos hasta que usted cierre de nuevo el fichero tecleando USE sin ningún nombre de fichero detrás, o con un nombre de fichero distinto. El coman- do CLEAR cierra también todos los ficheros abiertos pero puede borrar al mismo tiempo algunas cosas que usted no desee perder, como pueden ser las variables de memoria que haya definido.

El dBASE puede mantener abiertos dos ficheros a la vez, uno primario y otro secundario. Si teclea usted USE PERSONAL, PERSONAL se convierte automáti- camente en la base de datos primaria. Si teclea SELECT SECONDARY seguido por USE y el nombre del fiche- ro, entonces ese segundo fichero se convierte en su base de datos secundaria. A partir de ahí puede dirigirs usted a una u otra de las dos bases tecleando SELECT PRIMARY o SELECT SECUNDARY. El dBASE man- tendrá automáticamente su posición en cada fichero de forma separada, mientras su aplicación recorre lo distintos registros en ambos ficheros.

El comando USE es utilizado también para *cerr* un fichero de datos. Utilizar solamente USE, sin nin- gún nombre de fichero, cerrará todos los ficheros (pri- marios o secundarios) que estén abiertos en aquel mo- mento.

DISPLAYANDO DATOS

Cuando desee ver usted los datos de su base, el mando que debe utilizar es DISPLAY (mostrar). Si clea sólo DISPLAY, el dBASE le mostrará el registro el que haya quedado posicionado el dBASE la últi vez. En la siguiente sección le mostraremos cómo leccionar esta posición, pero antes necesitamos sa algo más acerca del DISPLAY.

Si desea ver el registro en curso, teclee DISPL

Si desea ver un registro en particular por su número (el cuarto, por ejemplo), puede teclear **DISPLAY RECORD 4**, y en su pantalla aparecerá el cuarto registro. El dBASE puede ser dejado también posicionado en este registro.

El formato del **DISPLAY** puede aparecer más bien como carente de estructura. Le mostrará a usted el registro cuyo número ha pedido y los campos de datos, uno tras otro. Los datos pueden parecerle extraños hasta que recuerde que los campos están llenos con blancos si los datos tecleados son más cortos que la longitud del campo. Esos blancos aparecerán en la pantalla como espacios vacíos de longitud supuestamente al azar. Si la estructura del registro es más larga que la longitud de una línea de su pantalla, el registro doblará a la línea siguiente o a las que sean necesarias.

También puede conseguir usted que le sean mostrados varios registros consecutivos tecleando por ejemplo:

```
DISPLAY NEXT 6
```

y verá en la pantalla el registro en curso y los cinco registros siguientes. Si desea toda una pantalla de registros, puede teclear:

```
DISPLAY ALL
```

y le será mostrada toda su base de datos, en grupos de quince registros cada vez. Puede pulsar cualquier tecla para trasladarse al siguiente grupo, o ^Q para finalizar el comando.

DISPLAY es en realidad una palabra de finalidad múltiple cuya misión es hacer que el dBASE le muestre algo. Puede usted **DISPLAY STRUCTURE** para que le muestra la estructura de la base de datos en **USE**. El Informe 4.1 le muestra un **DISPLAY** de la estructura de la base de datos que preparamos en el capítulo anterior.

Informe 4.1.

ESTRUCTURA PARA FICHERO: PERSONAL.DBF
NUMERO DE REGISTROS: 00018
FECHA ULTIMA ACTUALIZACION: 05/23/84
BASE DE DATOS PRIMARIA

FLD	NAME	TYPE	WIDTH	DEC
001	NUM:EMPL	C	004	
002	ANTIGDAD	C	006	
003	EXTN:TEL	C	004	
004	NOM:PILA	C	010	
005	APELLIDO	C	015	
006	CALLE	C	020	
007	CIUDAD	C	012	
008	ESTADO	C	002	
009	COD:POSTAL	C	005	
010	TEL:PART	C	011	
011	NO:SEG:SOC	C	009	
012	EXENC	N	002	
013	FCHA:NAC	C	004	
014	SUELDO	N	006	
015	HRAS:SEM	N	004	003
016	BRUT:SEM	N	007	001
017	BRUT:ANL	N	008	002
018	IMPFED:SEM	N	006	002
019	IMPFED:ANL	N	007	002
020	IMPEST:SEM	N	005	002
021	IMPEST:ANL	N	007	002
022	SEG:ANL	N	007	002
023	NET:SEM	N	007	002
024	NET:ANL	N	008	002
025	FECH:FIN	C	006	002
	TOTAL		00183	

También puede usar DISPLAY para hacer que el dBASE le muestre el *directorio* de ficheros de un drive de discos:

```
DISPLAY FILES ON B:
```

le mostrará el directorio del drive B. Puede utilizar también una *expresión* para limitar el tipo de ficheros que desea ver DISPLAYados, tecleando:

```
DISPLAY FILES LIKE *.DBF
```

Esto le mostrará todos los ficheros que posean el tipo de fichero .DBF, como el PERSONAL.DBF.

En los capítulos en que tratemos de la programación en dBASE examinaremos cómo elaborar algunas cantidades determinadas como *variables* para almacenar datos de texto o números sujetos a variación. También puede utilizar usted DISPLAY para que le sean mostradas esas variables. DISPLAY MEMORY le mostrará las variables en uso que esté utilizando en esos momentos, con sus valores o contenidos actuales.

LIST es otra palabra importante para que el ordenador nos muestre cosas. Si tecleamos:

```
USE PERSONAL
LIST
```

haremos que todos los registros del fichero PERSONAL vayan pasando sucesivamente por nuestra pantalla. También puede proporcionar usted expresiones para limitar los registros listados:

```
LIST FOR COD:POSTAL = "91007"
```

Del mismo modo, puede LIST STRUCTURE para obtener la estructura de la base de datos, y ver el directorio del disco con LIST FILES ON B.

El dBASE proporciona algunas palabras de utilidad para modificar los comandos display. Si añade

OFF a los comandos que nos muestran datos, los números de registro no serán exhibidos. LIST OFF mostrará el contenido de datos de los registros, pero sin los números de esos registros. El modificador NEXT puede ser utilizado para especificar un grupo de registros que sigan al registro en uso. LIST NEXT 5 le mostrará cinco registros a partir del registro en uso.

MOVIÉNDOSE ENTRE SUS DATOS

USE le posiciona automáticamente en el primer registro del fichero. Puede usted teclear el comando DISPLAY y ver ese registro. Pero a menudo necesitará ver otros datos distintos a los de ese primer registro. El dBASE posee algunos comandos para ayudarle a moverse en su base de datos. Algunos son comandos de *movimiento absoluto*, que lo trasladan a un lugar determinado. Otros corresponden a *movimientos relativos*: se trasladan cierto número de registros hacia adelante o hacia atrás a partir del lugar donde esté usted posicionado.

Los comandos GO o GOTO pueden llevarle hasta lugares específicos: GOTO TOP lo traslada al primer registro, GOTO BOTTOM al último. Si conoce usted el número de registro del dato que quiere ver, GOTO *n* es un comando que lo situará en un lugar específico de la base de datos: el registro *n*. Por ejemplo, GOTO 7 le posicionará en el séptimo registro. ¿Cómo saber esos números de registro? BROWSE, LIST o DISPLAY ALL son comandos que le permitirán examinar toda la base de datos buscando una información en particular. Una vez haya descubierto dónde está, puede GOTO al registro que le interesa y DISPLAY para que aparezca en pantalla, o más simplemente puede DISPLAY RECORD *n*.

Quizá lo que más frecuentemente necesite sea posicionarse en el siguiente registro para examinarlo. El comando de movimiento para esto es SKIP (salto). Teclear el comando le traslada al siguiente registro

También puede añadirle usted números *relativos* de movimiento para avanzar o retroceder un cierto número de registros. Si en estos momentos se halla posicionado en el registro 4 y teclea:

```
SKIP -2
DISPLAY
```

le será mostrado el registro #2. (Se supone que los números de SKIP son positivos, avanzando en consecuencia a lo largo del fichero, a menos que los preceda usted del signo menos. El signo "+" no es necesario.)

ESPECIFICANDO O LIMITANDO DATOS

A menudo deseará usted examinar más datos que los de un solo registro, pero menos que los de todo el fichero. Lo que necesita es pues algún método de limitar o seleccionar los datos que desea que le muestre el dBASE. El dBASE posee algunas capacidades realmente poderosas en este sentido. Vimos una de ellas en la sección anterior: LIKE, para mostrar un grupo de ficheros con nombres de fichero similares.

Una capacidad importante del dBASE es permitir el uso de *expresiones*. Una expresión es una ecuación o comparación, que se utiliza ya sea para calcular un valor o para especificarle algunos criterios al dBASE. Cuando busque usted datos, por ejemplo, puede utilizar una expresión o comparación para ayudar al dBASE a decidir si un registro en particular es o no el que desea ver.

Hay varios tipos de expresiones. Las aritméticas pueden incluir la suma, la resta, la multiplicación y la división:

```
DISPLAY FOR PQUANT + RQUANT =100
```

le mostrará todas las facturas con las sumas de las partes P y R ordenadas sobre 100.

Las expresiones de comparación incluyen igualdades o desigualdades, y mayor o menor que:

```
DISPLAY FOR FAC:TOTAL > 1000
```

mostrará todas las facturas superiores a 1.000$ (suponiendo que su base de datos tenga un campo FAC:TOTAL).

Las expresiones no están limitadas a los números en el dBASE. Puede trabajar usted también con cadenas de caracteres de texto. ¿Cómo hacerlo para unir caracteres? A la operación se la llama *concatenación*:

```
"abc" + "xyz"
```

es una expresión válida en dBASE (al igual que en otros lenguajes de ordenador), que le proporcionará la cadena de caracteres abcxyz. La expresión:

```
DISPLAY FOR APELLIDO > "JOHNSON"
```

le ofrecerá en pantalla todos los registros de PERSONAL con nombres alfabéticamente posteriores a Johnson.

Todo el mundo está familiarizado con las operaciones aritméticas estándar. La mayor parte de las personas están familiarizadas con las operaciones de comparación, llamadas técnicamente operaciones *relacionales*. Las operaciones *en cadena* –es decir, las operaciones con grupos de caracteres de texto– se hallan un poco más alejadas de la experiencia de mucha gente.

Hemos visto que puede usted unir cadenas por concatenación. Puede que desee hacerlo, por ejemplo para establecer un nombre completo a partir del nombre de pila y el apellido almacenados en cadenas separadas. Veamos cómo puede conseguirse esto.

Su base de datos incluye dos campos, el nombre de pila y el apellido. Recuerde que cada uno de ellos posee un número fijo de caracteres, que pueden ser más de los necesarios para su contenido. Los campos son

llenados con blancos si el nombre es más corto que el campo en el que está almacenado. Supongamos que su campo NOM:PILA de 10 caracteres contiene «JOHN» y el APELLIDO de 12 caracteres es «JONES». Si une usted los dos campos con la operación "+", obtendrá una cadena:

```
"JOHN      JONES       "
```

Si intenta la concatenación con el operador "–", que traslada los blancos de ambos campos al extremo de la derecha, obtendra:

```
"JOHNJONES             "
```

El dBASE le permite TRIM (recortar) los blancos sobrantes, pero esto elimina el blanco que debería aparecer entre el nombre de pila y el apellido. Así que hay una forma en que puede utilizar usted una expresión para concatenar los nombres de pila y los apellidos en el dBASE:

```
STORE TRIM (NOM:PILA) + " " + ;
  TRIM (APELLIDO) TO NOMBRE
```

Esta operación de cadena producirá en NOMBRE el nombre completo:

```
"JOHN JONES"
```

de la forma normal en que debe ser escrito, con un solo espacio entre nombre de pila y apellido.

El dBASE incluye algunas operaciones y valores que puede que no resulten familiares a los recién llegados al mundo del ordenador. Implican valores que son o ciertos o falsos, o sí o no, y operaciones que producen tales valores. Por ejemplo, mientras estamos examinando un fichero de datos, una persona puede preguntar: «¿Has terminado ya con el último registro?» En lenguaje de ordenador, esto puede traducirse como:

```
IF EOF
```

que devuelve el valor de cierto o sí si el dBASE ha llegado ya al final del fichero, o de falso o no en caso contrario. El comando IF del dBASE ejecuta la orden que sigue a continuación tan sólo si la expresión resultante tiene un valor de cierto. Puesto que no cierto equivale a falso y viceversa, el comando

```
IF .NOT. EOF
```

es una forma habitual de especificar que debe seguirse con el procesado si aún quedan datos por procesar. Por ejemplo, si está utilizando usted el dBASE para preparar facturas a partir de un archivo determinado, puede utilizar este comando antes de sus comandos de procesado de las facturas.

Puede que esas operaciones lógicas sean nuevas para usted. Son .AND., .OR. y .NOT. .AND. une otras dos operaciones lógicas, y devuelve un valor de cierto *solamente* si ambas operaciones son ciertas. Por ejemplo:

```
DISPLAY FOR NOM:PILA = "ROGER" .AND.;
  APELLIDO = "JOHNSON"
```

mostrará solamente el registro del empleado ROGER JOHNSON, no los registros de SAM JOHNSON o de ROGER SMITH. La operación lógica .OR. devolverá un valor de cierto si *una* de las dos operaciones lógicas que une es cierta:

```
DISPLAY FOR NOM:PILA = "ROGER" .OR.;
  APELLIDO = "JOHNSON"
```

mostrará los tres: ROGER SMITH, ROGER JOHNSON y SAM JOHNSON.

.NOT. es un inversor; convierte lo cierto en falso viceversa.

El dBASE posee un operador de cadenas lógica muy poderoso llamado $. Esta operación comprueb

para ver si una cadena se halla contenida dentro de otra. Por ejemplo, la operación

"ELM"$DIRECCION

devolverá un valor de cierto para cada dirección de CALLE ELM y AVENIDA ELM. Por supuesto, dará también un valor de cierto para CHELMSFORD o cualquier otra palabra que contenga la secuencia de letras ELM.

La Tabla 4.1 muestra los tipos de operadores aritméticos, relacionales, de cadena y lógicos utilizados en el dBASE.

Tabla 4.1.

=	igual a
>	mayor que
>=	mayor que o igual a
<	menor que
<=	menor que o igual a
<> o	desigual
+	suma aritmética o concatenación de cadenas (unión)
-	resta aritmética o concatenación de cadenas con blancos sobrantes trasladados al extremo de la derecha
*	multiplicación
/	división
()	paréntesis para agrupar subexpresiones
.AND.	devuelve cierto si ambos lados son ciertos
.OR.	devuelve cierto si uno de los dos lados es cierto
.NOT.	negación: convierte cierto en falso, falso en cierto
$	busca una cadena dentro de otra

SORTANDO SUS DATOS

Una de las tareas más útiles que puede hacer el dBASE es clasificar sus datos en el orden que usted desee. Por supuesto, no deseará usted que se tomen en cuenta todos los datos

cuando se efectúe esta clasificación, sino tan sólo el campo más importante de la lista. Ese campo es llamado el campo *clave* sobre el que se efectúa la clasificación. El dBASE puede efectuar clasificaciones tanto numéricas como alfabéticas, y tanto en orden ascendente como descendente.

Hemos mencionado antes dos listas clasificadas útiles que puede desear usted para su base de datos de empleados. Veamos cómo conseguirlas.

Su base de datos PERSONAL contiene información de la fecha de nacimiento de cada empleado. Ahora es el momento de producir esa lista que mencionamos para que el director pueda utilizarla a fin de enviar sus felicitaciones de cumpleaños. Por supuesto, deseamos la lista en orden de fechas del 1 de enero al 31 de diciembre. Luego, cuando se llegue a cada fecha, puede ser tachada y la siguiente de la lista será la vigente. Para clasificar los datos para esa lista, teclearemos:

```
USE PERSONAL
SORT ON FCHA:NAC TO LISTFCHN
```

El dBASE producirá una *nueva base de datos* llamada LISTFCHN.DBF, con todos los registros que figuran en PERSONAL, pero ordenados desde la primera fecha de nacimiento del año a la última. (En la clasificación se supone siempre un orden ascendente si no se especifica lo contrario.) ¿Cuánto tiempo requerirá esta clasificación? Mi sistema empleó 1,4 segundos para crear la estructura de la nueva base de datos, clasificar los dieciocho registros y almacenarlos en una nueva base de datos. Observe que PERSONAL sigue intacta tras la clasificación, en el mismo orden que estaba antes de emplear el comando SORT. Ahora disponemos de dos bases de datos, ambas con la misma información, pero en un orden diferente útil para dos finalidades distintas.

Observe también que no vamos a examinar ahora la forma de imprimir esta lista de cumpleaños. Nos ocuparemos de ello en el siguiente capítulo. Pero para

asegurarnos de que los datos están ahí, si sigue usted este texto trabajando simultáneamente con el dBASE en su ordenador, teclee USE LISTFCHN y BROWSE para ver los registros en orden de cumpleaños.

También puede que deseemos un listín telefónico. De hecho, nos interesan *dos* listines telefónicos. Uno reflejará a los empleados por el apellido, el nombre de pila y la extensión. El otro reflejará a los empleados por orden numérico de extensión telefónica.

Podremos completar rápidamente una base de datos para la primera lista tecleando:

```
USE PERSONAL
SORT ON APELLIDO TO NOMBEXT
```

Una base de datos estructurada para proporcionarnos la segunda lista es creada así:

```
USE PERSONAL
SORT ON EXTN:TEL TO EXTNOMB
```

De nuevo, toma mas tiempo teclear los comandos que la clasificación en sí, para una base de datos de este tamaño. Más adelante veremos cómo imprimir estas dos listas.

Hay un punto importante a recordar respecto a la clasificación en dBASE, que ya hemos mencionado antes. Los caracteres se hallan almacenados en la mayoría de los ordenadores en un código llamado ASCII: el Código Estándar Americano para el Intercambio de Información. En este código, las letras mayúsculas poseen un código inferior a las minúsculas. El dBASE clasifica en orden ASCII, no sólo alfabéticamente. Los nombres jones y Smith aparecerán en orden inverso, Smith antes que jones, debido a que cualquier letra mayúscula aparecerá antes en la estructura de clasificación que cualquier letra minúscula con la que sea comparada. Cualquier palabra que empiece con mayúscula será clasificada antes en el orden alfabético que cualquier palara que empiece con minúscula. Del

mismo modo, la palabra BRAHMS será clasificada antes que Bach, y la abreviatura del estado de Colorado COL vendrá antes que la del de California Cal. Si ha tecleado usted sus datos de una forma coherente (todo mayúsculas, todo minúsculas, sólo mayúsculas las iniciales, etc.) en todos los campos correspondientes de cada registro, no tendrá ningún problema en este aspecto.

CLASIFICACIÓN EN CLAVE MÚLTIPLE

Ahora veamos algo más respecto a las clasificaciones. Cualquier clasificación, ya sea numérica o alfabética, es realizada únicamente sobre un campo: la *clave* que le proporciona usted al comando SORT. Ésta es una característica del dBASE: puede clasificar solamente sobre un campo a la vez. ¿Puede esto crear problemas?

Si ha efectuado usted la clasificación por APELLIDO descrita más arriba, observará que los dos empleados llamados Ransom están juntos, pero el registro de Roger Ransom está clasificado antes que el de Emma Ransom. Puesto que no hay ninguna clasificación por nombres de pila, los dos Ransom son listados en el orden en que figuran en la base de datos original. Roger estaba primero, así pues aparece primero en la base de datos clasificada. Para evitar esto, sería muy útil que la base de datos pudiera ser clasificada simultáneamente sobre dos (o más) claves. Podemos darle instrucciones al ordenador para que clasifique al mismo tiempo sobre el apellido y sobre el nombre de pila, y entonces Emma irá delante de Roger, como corresponde. La mayor parte de los fabricantes de programas para ordenador ponen constantemente al día sus productos, y Ashton Tate, el fabricante del dBASE, no es una excepción. Una versión futura del dBASE ofrecerá indudablemente la clasificación en clave múltiple.

Por ahora, sin embargo, tenemos que utilizar una

técnica distinta para conseguir el mismo resultado. Si clasificamos primero el fichero **PERSONAL** sobre NOM:PILA a un fichero intermedio, y luego clasificamos ese fichero intermedio sobre **APELLIDO**, obtendremos el orden correcto para la lista de extensiones telefónicas. Esto ocurrirá porque Emma irá delante de Roger en el listado intermedio.

En consecuencia, para una clasificación en clave múltiple, efectúe una serie de clasificaciones sucesivas. *Utilice primero la clave menos significativa*, luego la siguiente menos significativa, y así sucesivamente hasta clasificar la clave más importante *la última*. Si tecleamos:

```
USE PERSONAL
SORT ON NOM:PILA TO TEMP
USE TEMP
SORT ON APELLIDO TO NOMBEXT
```

obtendremos los datos definitivos para el listín de extensiones telefónicas en el orden que deseamos, clasificados a la vez según el apellido y el nombre de pila. El fichero (o ficheros) intermedio puede ser desechado.

INDEXando SUS DATOS

Puede utilizarse otra técnica, además del SORT, para clasificar una base de datos. ¿Cuántas veces ha utilizado usted un índice para encontrar algo en un libro? El dBASE puede construir un índice para un fichero de base de datos si usted se lo pide. El comando SORT mueve realmente los datos dentro de la base de datos, creando un orden distinto de los registros que existían antes de la clasificación. INDEX deja la información de la base de datos tal como estaba, pero crea un índice clasificado. Este índice permite una localización muy rápida de los datos, mucho más rápida que buscar el mismo dato a través del fichero.

Si LISTa el fichero de base de datos antes de SORT,

verá los registros en el orden en que fueron entrados. Tras el SORT, los registros se dispondrán en el orden de la clasificación indicada. Sin embargo, si INDEXa la base de datos original y luego la lista, su orden puede ser distinto, dependiento de si USE o no el fichero INDEX.

```
USE PERSONAL
LIST
```

le dará el orden original.

```
USE PERSONAL
SORT ON APELLIDO TO TEMP
USE TEMP
LIST
```

le dará los registros de la base de datos, clasificados ahora por los apellidos. Sin embargo,

```
USE PERSONAL
INDEX ON APELLIDO TO NOMFICH
```

le dará, cuando pida LIST, los registros en su orden original. Y si ordena:

```
USE PERSONAL INDEX NOMFICH
LIST
```

sus ficheros le serán mostrados en orden alfabético.

A diferencia de SORT, puede construirse un índice sobre más de una clave. Puede, por ejemplo, indexar el fichero PERSONAL basándose en apellidos y nombres de pila a la vez:

```
USE PERSONAL
INDEX TO APELLIDO + NOM:PILA TO NOMINDEX
```

El dBASE construirá muy rápidamente un fichero llamado NOMINDEX que facilitará la rápida locali-

zación de los datos en los archivos no clasificados de PERSONAL, basada tanto en el campo APELLIDO como en el NOM:PILA. El fichero PERSONAL en sí no resulta cambiado, pero usándolo a través de NOMINDEX conseguimos el mismo efecto que si tuviéramos los datos del fichero clasificados de acuerdo con el campo que hayamos elegido para el índice.

Para usar el fichero PERSONAL a través del índice tiene que teclear:

```
USE PERSONAL INDEX NOMINDEX
```

Ahora, todas las peticiones de información basadas en nombres pasarán a través del archivo indexado.

Tenga en cuenta que una vez construido un fichero índice, no es obligatorio que USE ese índice como entrada única al fichero de base de datos. Los datos pueden seguir siendo localizados y recuperados por cualquiera de los comandos secuenciales de búsqueda. Sin embargo, si ha empleado el USE con un nombre de fichero INDEX en un fichero, entonces todas las referencias al fichero irán vía índice. Si, por ejemplo, está usando usted el fichero PERSONAL vía NOMINDEX, no podrá buscar un registro basado en la antigüedad en el empleo.

Un fichero de base de datos puede tener muchos ficheros índice diferentes referidos a él, lo cual ofrece el mismo resultado que crear el mismo número de ficheros de datos clasificados, pero usando mucho menos espacio de almacenamiento. El dBASE contiene un comando, SET, que tiene muchas variaciones. Una de ellas es utilizada para cambiar ficheros índice. Si emplea usted el USE con un fichero de datos a través de un fichero índice particular, puede cambiar de fichero índice tecleando:

```
SET INDEX TO nuevonombrefichero
```

Esto cerrará el primer fichero índice y abrirá el segundo.

En teoría, cuando se están usando ficheros de datos con un índice, al cerrar el fichero de datos tecleando USE se deberían cerrar también todos los ficheros índice que se hayan seleccionado. Sin embargo, he oído hablar de casos en los que al parecer esto no ocurre. Un método más conservador de asegurar el cierre podría ser teclear SET INDEX TO sin ningún nombre de fichero para cerrar el fichero índice, luego teclear USE para cerrar el fichero de datos.

Hay alguna ventaja y alguna desventaja secundarias en usar un fichero índice. La ventaja es que, si está añadiendo usted registros a la base de datos y ha señalado USE para el fichero índice (o todos los índices apropiados, si hay varios), el fichero o ficheros índice serán puestos automáticamente al día a medida que son añadidos nuevos registros. La desventaja es que, si ha añadido usted datos sin haber marcado USE para el fichero índice, el índice y la base de datos ya no se corresponderán. Tendrá que volver a indexar la base de datos.

BUSCANDO DATOS

Ahora tenemos que empezar a trabajar con métodos para conseguir que todos los datos almacenados en el ordenador sean útiles. Lo primero que tenemos que hacer es ser capaces de encontrar un dato en particular. Iniciamos eso en la sección anterior, cuando examinamos el indexado. Ahora introduciremos el comando LOCATE.

Supongamos que está buscando usted una determinada hoja de papel en una carpeta repleta de papeles. Si los papeles estuvieran ordenados al azar, empezaría usted con el primer papel, y decidiría si era el que deseaba. Si no lo era, lo dejaría a un lado y pasaría al siguiente.

Puede hacer lo mismo en el dBASE. Puede ir usted al principio («top») del fichero de datos (recuerde que USE nombrefichero hace eso por usted) y examinar el

primer registro. Si el contenido del campo que está examinando no es el que desea, SKIP (salte) al siguiente registro. Sin embargo, eso requiere un montón de comandos para conseguir lo mismo que puede hacer LOCATE por usted en una sola línea.

LOCATE es un tipo de comando de búsqueda que encuentra un dato en particular. De hecho, encuentra la primera vez en que aparece dicho dato. Si desea encontrar usted el registro personal de su empleado MOZART, puede teclear:

```
LOCATE FOR APELLIDO = "MOZART"
```

LOCATE encontrará el registro, que puede usted llamar con DISPLAY.

Observe que el nombre MOZART es colocado entre comillas. En dBASE (como en muchos otros lenguajes de ordenador), así es como define usted un *literal*, un grupo de caracteres invariables que deben ser buscados exactamente igual a como han sido tecleados.

No necesita utilizar LOCATE cuando se halle utilizando una base de datos a través de un fichero índice que se halla indexado sobre cualquier otro campo distinto a aquél en el que está interesado.

FIND es un localizador muy rápido cuando se usa un fichero índice. Cuando esté buscando información en un fichero de datos indexados, incluso uno almacenado en su diskette, FIND le proporcionará lo que busca (si existe) en uno o dos segundos. La cadena de caracteres que quiere usted encontrar no tiene que ser siquiera un nombre completo. Puede teclear usted tan sólo lo suficiente del campo que está buscando como para distinguir el dato de ese campo de cualquier otro dato. Por ejemplo, para encontrar el registro que contiene el apellido MOZART (suponiendo que haya construido un índice de apellidos), puede teclear:

```
USE PERSONAL INDEX APELLIDO
FIND MOZ
DISPLAY
```

y el registro del Sr. Mozart aparecerá en su pantalla.

Observe que estas técnicas son utilizadas para encontrar y mostrar registros individuales. El dBASE puede hacer fácilmente esa tarea, pero es mucho más potente que eso. Muy a menudo deseará usted informes de su base de datos que cubran varios o todos los registros. Veremos este tema a continuación.

5. EL dBASE II COMO GENERADOR DE INFORMES

Con LOCATE o FIND y DISPLAY puede hacer que los registros individuales que desea sean extraídos de la base de datos. Mantener todos esos datos disponibles para una rápida recuperación como registros individuales es sólo parte de las capacidades del dBASE. Los datos existen solamente por la información que pueden proporcionarle: no sólo los datos en sí contenidos en cada registro, sino resúmenes e informes preparados según sus especificaciones. El dBASE puede proporcionarle muchos tipos distintos de informes. Esos informes pueden ser enviados a la pantalla, a la impresora de su ordenador, o a ambos.

Algunos de los informes que puede preparar el dBASE son muy directos, y necesitan poco formateo. Otros requieren una programación especial, quizá debido a algunos cálculos necesarios o a problemas especiales de recuperación. En este capítulo examinaremos primero la utilización del dBASE para informes interactivos con los comandos SUM y COUNT, entre otros. Luego terminaremos efectuando bases de datos resumen con TOTAL, y examinando las capacidades del REPORT incluido en el dBASE. En el siguiente capítulo veremos la muy potente capacidad de produir informes formateados a la medida y que implican extensa manipulación y cálculo de datos.

Recuede que la mayor parte de los sistemas operativos de ordenador poseen algún medio de enviar el

output de la pantalla a la impresora. Por ejemplo, en el CP/M, puede usted pulsar control-P (^P), y cualquier texto enviado a partir de entonces a la pantalla será enviado también a la impresora. Pulsar de nuevo ^P detiene esta acción. En el ordenador personal IBM que utiliza el sistema operativo PC-DOS de IBM, pulsar ^PrtSc produce el mismo resultado.

GENERACIÓN INTERACTIVA DE INFORMES CON EL dBASE II

Muchos de los informes que puede desear usted del dBASE son relativamente simples, de modo que puede que ni siquiera desee un *report* escrito. El dBASE puede operar en modo interactivo, lo cual significa que usted establece una especie de diálogo con el ordenador que está ejecutando el dBASE. Usted teclea los comandos, y el dBASE va en busca de la información que usted necesita.

¿Cuáles son algunas preguntas sencillas de su base de datos PERSONAL que puede hacerle al dBASE para que le responda? Una puede ser: «¿Cuántas personas trabajan aquí?» Si tiene usted un registro por empleado y borra los registros cuando uno de ellos se despide, responderla es fácil. Simplemente vea cuál es el número del registro del último empleado en la base de datos:

```
USE PERSONAL
DISPLAY STRUCTURE
```

La estructura que aparezca en pantalla le dirá cuántos registros hay en la base de datos.

Pero supongamos que desea usted saber cuántas personas llevan trabajando con usted más de dos años. Si la fecha de hoy fuera el 10 de agosto de 1984, entonces esa fecha podría ser representada como 840810 en el formato que he recomendado antes. Una antigüedad del 820810 sería la fecha de hace dos años. Si le decimos al ordenador:

```
USE PERSONAL
COUNT FOR ANTIGDAD < "820810"
```

el dBASE revisará los registros de sus empleados, contando todos aquellos cuya fecha de alta se remonta a más de dos años, y muy rápidamente le presentará su respuesta:

```
COUNT = 10
```

Hay muchas preguntas a las que el dBASE puede responder de forma interactiva acerca de sus datos. Utilizando expresiones para definir los datos que desea, más los comandos COUNT, SUM y DISPLAY, puede conseguir usted que el dBASE de su ordenador le proporcione un montón de información útil. ¿Desea saber cuánto sube su nómina acumulada de este año hasta el día de hoy, excluidos los propietarios? Teclee:

```
USE PERSONAL
SUM TOT:ANL FOR APELLIDO <> "RANSOM"
```

El dBASE recorrerá de nuevo todos los registros, sumando los salarios acumulados de cada empleado excepto los Ransom. En un segundo o así, el dBASE le informará del total. En este caso es 0,00 puesto que todavía no hemos entrado ningún dato ni hecho ningún cálculo que altere el cero inicial que pusimos en ese campo.

Casi todos los comandos que estamos examinando en este manual son utilizables directamente a partir del teclado de su ordenador. Puede utilizarlos usted para extraer datos o resúmenes de datos para cubrir sus necesidades del momento. Por ejemplo:

```
DISPLAY FOR FCHA:NAC > "0331" .AND.;
  FCHA:NAC < "0501"
```

listará en su pantalla a aquellos empleados que hayan nacido en abril. Esos breves comandos pueden ser

muy útiles para peticiones de datos por una sola vez. Más adelante, en este mismo capítulo y en el capítulo siguiente, veremos cómo conseguir datos de una forma repetitiva o más compleja.

CREANDO BASES DE DATOS RESUMEN

Supongamos que ha utilizado usted el dBASE para controlar las ventas y preparar facturas. A final de mes, quiere saber el total de ventas a cada cliente, o el total de ventas correspondiente a cada vendedor de su empresa. Supondremos que su base de datos de ventas, VENTAS, está definida de tal modo que cada registro (correspondiente a cada venta) contiene la fecha de la venta como FECHA, el nombre del cliente como COMPRADOR, el nombre del vendedor como VENDEDOR, y la cantidad total de la venta como IMP:FAC.

El dBASE posee un comando, TOTAL, que recorrerá toda su base de datos e irá sumando información, almacenando la suma en una nueva base de datos que contiene solamente esa información resumen. La base de datos maestra tiene que ser SORTada o INDEXada sobre la clave usada para el resumen. Así pues, antes de extraer los totales, prepararemos dos ficheros índice.

```
USE VENTAS
INDEX ON COMPRADOR TO COMINDEX
INDEX ON VENDEDOR TO VENINDEX
```

Ahora crearemos una base de datos resumen para el informe mensual de agosto de 1984 de las compras de cada comprador.

```
USE VENTAS INDEX COMINDEX
COPY TO COMPREP STRUCTURE FIELDS;
   COMPRADOR,IMP:FAC
TOTAL ON COMPRADOR TO COMPREP FIELDS;
   IMP:FAC FOR FECHA > "840731"
            .AND. FECHA < "840901"
```

Primero hemos creado la estructura de una nueva base de datos que contiene solamente los dos campos de VENTAS en los que estamos interesados: COMPRADOR e IMP:FAC. Si no hiciéramos eso, el comando TOTAL habría copiado toda la estructura de la base de datos VENTA en la base de datos REPCOMP, y perderíamos un montón de espacio. Luego, le hemos dicho al dBASE que totalice los campos IMP:FAC en todos los registros de la base de datos de ventas, pero que lo haga por COMPRADOR (cliente), y tan sólo para el mes de agosto de 1984. La nueva base de datos, REPCOMP, contendrá solamente un registro por cliente, mostrando el total de compras efectuadas por ese cliente en agosto.

```
USE VENTAS INDEX VENINDEX
COPY TO REPVEND STRUCTURE FIELDS;
  VENDEDOR,IMP:FAC
TOTAL ON VENDEDOR TO REPVEND FIELDS;
  IMP:FAC FOR FECHA > "840731"
          .AND. FECHA < "840901"
```

La base de datos VENTAS original no ha sido alterada de ninguna forma. Hemos utilizado el método INDEX para crear dos formas distintas de contemplar la base de datos de una forma clasificada, sin clasificarla realmente. El comando TOTAL necesita este enfoque preclasificado para funcionar. TOTAL no puede funcionar en la base de datos VENTAS en bruto.

Para mostrar los datos resumen que acabamos de preparar, podemos ordenar LIST o DISPLAY sobre la base de datos resumen. El listado de los datos de REPCOMP nos proporcionará una relación de clientes y la cantidad comprada por cada uno de ellos durante el mes. El listado de los datos de REPVEND nos dará una compacta relación de cuánto ha vendido cada vendedor en el mes de agosto.

EL dBASE COMO REPORTER

Hemos visto tres formas en que puede conseguir usted datos de una forma más o menos no estructurada: DISPLAY o LIST, peticiones interactivas con SUM y COUNT, y resúmenes TOTAL. En esta sección examinaremos un modo más formal de obtener informes utilizando el comando REPORT del dBASE.

Un vez haya seleccionado su base de datos con USE, puede teclear usted REPORT para crear un formulario para un informe que necesite más de una vez. El dBASE le conducirá mediante un diálogo interactivo a elaborar ese formulario. En primer lugar el dBASE le pedirá un nombre para el informe; en nuestro ejemplo eligiremos LISTEL. Luego, tras darle la oportunidad de fijar opciones tales como márgenes y líneas por página, el dBASE le pedirá que establezca cada columna del informe: una columna de encabezamiento, si procede, con el título de los campos que la formen. Para nuestro formato LISTEL queremos mostrar NOM:PILA, APELLIDO y EXTN:TEL. A fin de que no resulte cortado ningún dado, elegiremos la misma anchura de las columnas que los campos en la base de datos. Sin embargo, puesto que los encabezamientos de las columnas son impresos con la misma anchura que la columna, incrementaremos la anchura de EXTN:TEL para el informe en cinco.

Una vez especificado el formato, el dBASE lo salva como un fichero bajo el nombre LISTEL.FRM. Ahora cada vez que necesitemos un nuevo listín telefónico podemos teclear REPORT FORM LISTEL, y el dBASE preparará el listín para nosotros.

¿Qué base de datos debemos USE? En el último capítulo, estudiando los comandos SORT e INDEX, hemos clasificado PERSONAL según los apellidos en una nueva base de datos, EXTNOMB. Si tecleamos USE EXTNOMB y luego ejecutamos nuestro informe, obtendremos el listín en orden alfabético según el apellido. Sin embargo, para animarle a utilizar el método INDEX en vez del SORT, vamos a sugerirle para esta pe

ueña compañ a que el listín telefónico sea preparado obre los nombres de pila. Así pues, teclee:

```
USE PERSONAL
INDEX ON NOM:PILA TO TFNOS
USE PERSONAL INDEX TFNOS
REPORT FROM LISTEL TO PRINT
```

el dBASE hará que la impresora prepare nuestro lis- in de extensiones telefónicas. El Report 5.1 muestra el utput.

La lista anual de cumpleaños puede ser preparada gual de fácil. Teclear REPORT nos permite especifi- ar otro formato, esta vez uno con sólo los campos CHA:NAC, APELLIDO y NOM:PILA. Ese informe pre- entará el aspecto del Report 5.2. Las líneas de coman- os que producen este informe son:

```
USE PERSONAL INDEX LISTCUMP
REPORT FROM FCHA:NAC PLAIN TO PRINT
```

Observe que meses y días son mostrados de la mis- na forma en que se hallan almacenados en los datos: 216 para el 16 de diciembre, por ejemplo. En la si- uiente sección veremos una forma de conseguir que el BASE prepare una forma distinta de lista de cum- leaños, con las fechas traducidas a algo un poco más egible.

Pueden prepararse otros *reports*, para bases de da- os ventas/facturación, por ejemplo. REPORT posee na capacidad matemática limitada a peparar subto- ales y totales para distintos campos de datos. En el iguiente capítulo, dedicado a la programación, exa- minaremos el utilizar las capacidades de programa- ión del dBASE para formatear informes sin otras li- nitaciones más que el tamaño del papel.

Hay disponibles muchas opciones con el comando REPORT. Si insertamos PLAIN en la línea del coman- do haremos que el dBASE omita la fecha y número de página en el principio del informe, quizá para incluir-

Report 5.1.

PAG. NO. 00001
05/25/84
EXTENSIONES TELEFONICAS EMPLEADOS

NOM. PILA	APELLIDO	EXTENSION
ANNAMARIE	OETTINGER	106
ARNE	SVENSON	113
ARNOLD	PITTSINGER	117
CEASAR	FRANCK	115
EMMA	RANSOM	102
GEORGE	HANDLE	103
HANNAH	LORE	114
JIMMY	JOHNSON	109
JOE	HIDEN	107
JOHN	BOCK	105
JOHN	BRAMZ	108
JUAN	OCHOA	117
LOU	BAYTOVEN	114
MARIANNE	OSWALDO	112
RENATE	FUSBANCK	105
ROGER	RANSOM	100
THELMA	ROSEKRANZ	101
WILLIE	MOZART	110

lo en un documento de proceso de textos. Puede inclu solamente unos datos específicos en el informe aña diendo una expresión en la línea de comando. Puest que el dBASE puede expulsar una página de la impr sora antes de empezar el informe, puede usted SE EJECT OFF para evitarlo. El omitir la frase TO PRIN en el comando imprimirá su informe en la pantall una buena idea cuando estemos intentando algo nuev

Report 5.2.

LISTADO DE CUMPLEAÑOS

CUMP	APELLIDO	NOMB. PILA
0118	PITTSINGER	ARNOLD
0127	MOZART	WILLIE
0210	OSWALDO	MARIANNE
0223	HANDLE	GEORGE
0319	ROSEKRANZ	THELMA
0321	BOCK	JOHN
0331	HIDEN	JOE
0421	JOHNSON	JIMMY
0507	BRAMZ	JOHN
0530	SVENSON	ARNE
0704	FUSBANCK	RENATE
0728	LORE	HANNAH
0930	OCHOA	JUAN
1017	OETTINGER	ANNAMARIE
1127	RANSOM	EMMA
1210	FRANCK	CEASAR
1216	RANSOM	ROGER
1216	BAYTOVEN	LOU

6. EL dBASE II COMO PROGRAMADOR

En los dos últimos capítulos hemos introducido el concepto de *comandos* y cómo utilizarlos para conseguir que el dBASE haga lo que usted desea. Hemos supuesto que se halla usted sentado ante su ordenador, tecleando esos comandos en el teclado uno a uno y viendo los resultados en la pantalla o la impresora. Esto es llamado el modo *interactivo* del dBASE; el ordenador y su programa dBASE están ahí para hacer todos los trucos que usted les indique, uno tras otro.

Muchas de las tareas que deseará hacer usted son repetitivas; deseará hacerlas más de una vez. En ese aspecto, el dBASE es realmente maravilloso. Puede usted teclear una larga lista de comandos en un fichero, y decirle al dBASE que *ejecute* ese fichero cada vez que usted desee realizar esa tarea en particular. La larga lista de USE, FIND, DISPLAY, STORE y otros versátiles comandos es traspasada del fichero al ordenador; el dBASE los ejecuta uno a uno del mismo modo que lo haría si usted se los fuera tecleando. La única diferencia es la cantidad de tiempo que se ahorra usted.

El dBASe llama a esto un fichero de comandos, pero en realidad es un programa de ordenador escrito en el lenguaje de comandos del dBASE. Eso es lo que veremos en este capítulo: escribir programas dBASE. Si nunca ha escrito usted un programa de ordenador antes, no se preocupe. No estamos hablando del FORTRAN, BASIC o programas en lenguaje máquina. El

lenguaje dBASE es un grupo de palabras en inglés con claros significados. Construir un programa a partir de ellas es una simple extensión de teclear los comandos en sí, uno tras otro. Sin embargo, al elaborar un fichero de comandos, aprenderá de una forma relativamente fácil las bases de la programación de ordenadores.

ALGO DE PROGRAMACIÓN BÁSICA

Tenemos que apartarnos un momento del tema específico del dBASIC para examinar algunos conceptos básicos sobre programación que necesita comprender usted.

VARIABLES

La mayor parte de la programación utiliza el concepto de *variables*. Una variable es un *nombre* para una localización de almacenaje, dentro de la cual puede almacenar usted y desde la que puede recuperar una información cambiante. Ya que hemos estado trabajando con nóminas, probablemente necesitemos una variable llamada NOM:NET. Puede usted crear una variable simplemente utilizando su nombre cuando establezca su valor inicial:

```
STORE 0 TO NOM:NET
```

crea la variable NOM:NET y establece su valor inicial en 0. Ahora, cada vez que hagamos cálculos sobre la nómina de un empleado, podremos almacenar el resultado en NOM:NET. Los programadores usan a veces trucos para separar o definir mejor las cosas. Uno de tales trucos es nombrar las variables haciendo que empiecen con la letra M, lo cual a simple vista les muestra que son variables de memoria y no campos de base de datos. Así que llamaremos a nuestra variable MNOM:NET.

El dBASE le permite disponer de sesenta y cuatro variables a la vez. Aunque esto suena a mucho, los programas grandes necesitan a menudo cuidadosos cálculos para evitar alcanzar el límite. Cuando ha terminado usted con una variable, puede desembarazarse de ella, haciendo así sitio para otra variable que pueda necesitar, utilizando RELEASE. Teclear (o incluir en un programa) RELEASE MNOM:NET borrará esa variable de la memoria. Si ha terminado usted con todas las variables utilizadas en el programa, puede dar la instrucción RELEASE ALL. De mismo modo, teclear RELEASE ALL LIKE M* borrará todas las variables de la memoria que empiecen con la letra M. Esto es muy útil si alguna vez profundiza a varios niveles de programas: que un programa llame a otro, el cual a su vez llame a otro, y así sucesivamente. Si la letra de principio para las variables creadas en cada nivel de programa es elegida de forma única, entonces el comando RELEASE ALL LIKE... puede ser incluido como parte de los comandos de salida de cada segmento de programa, para borrar únicamente las variables creadas a ese nivel de programa, sin tocar las variables de otras áreas del mismo. Por ejemplo, puede decidir usted que el programa principal cree variables que empiecen con A, que el siguiente nivel cree variables con B, y así sucesivamente. Si hace usted eso, recuerde al nombrar cualquier variable que un programa le devolverá al programa que lo llamó con el prefijo del programa *al que se llama*, no del programa *llamado*, a fin de no ser borrados (por el comando RELEASE) inadvertidamente. Veremos ejemplos de esta técnica más adelante, en nuestro programa de nómina.

Literales

Otros conceptos que es preciso comprender es la idea de *literales*. Aunque esto le suene a nuevo, ya ha visto ejemplos de ello. Un literal es un texto que desea usted que sea manejado exactamente tal como está. Se

diferencia por ejemplo de un nombre variable, que es tratado como una referencia a un valor almacenado en algún lugar. Los literales se muestan enmarcados en apóstrofes o en comillas. En el capítulo 4 vimos la forma de recuperar datos clasificados por el código postal. Recordando que el código postal se halla almacenado en nuestra base de datos en forma de caracteres y no de números, tecleamos:

```
LIST FOR COD:POSTAL = "91007"
```

El código postal "91007" es un literal, almacenado como los caracteres separados "9", "1", "0", "0" y "7", y no como el número 91007. Cada vez que desee usted utilizar un texto específico en el dBASE, tiene que hacerlo como un literal. Si teclea:

```
STORE "JOHN" TO MNOM:PILA
```

los caracteres JOHN serán almacenados allí. Pero si teclea:

```
STORE JOHN TO MNOM:PILA
```

el dBASE buscará una variable o un campo llamado JOHN, cuyo *contenido* debe ser almacenado en MNOM:-PILA. Suponiendo que el dBASE no podrá encontrar lo que busca, todo lo que obtendrá usted será un mensaje de error.

Probablemente uno de los errores más frecuentes al iniciarse uno en la programación dBASE sea olvidarse los apóstrofes o comillas que señalan los literales. Su marca para abrir y cerrar tiene que ser siempre la misma: utilice el apóstrofe o las comillas en ambos lados, no uno en cada. Si tiene que emplear un apóstrofe o comillas dentro del literal, como por ejemplo en el nombre de un cliente, *John's Bar*, utilice entonces la otra marca para señalar el literal: "John's Bar".

Un programa

Un programa dBASE es simplemente una secuencia de comandos, que el ordenador lee y ejecuta. En vez de tener que teclearlos, se hallan almacenados en el ordenador. El dBASE los va tomando de uno en uno, interpretando la acción que usted desea que haga y realizándola. Tomémonos un momento para ver los pasos que debe seguir usted para escribir un programa dBASE.

Normalmente cada programa requiere algún input del operador. Nuestro programa de nómina, por ejemplo, necesita saber las horas trabajadas por cada empleado durante la última semana. Puesto que este input debe ser completado antes de poder realizar el balance del trabajo, el primer paso en un programa dBASE es diseñar los inputs. Normalmente el programa presenta un *menú* de actividades que puede realizar por el operador, y una o más *pantallas* de prompts para obtener la información necesaria.

Tras recibir los datos, el dBASE seguirá adelante y efectuará las tareas asignadas. El resultado será habitualmente alguna forma de output: un informe, los cheques ya impresos, o algo parecido. Así que el segundo paso es diseñar cuál será el aspecto del output.

El último paso es definir la secuencia real de comandos necesaria para transformar el input y los datos almacenados en el output deseado.

Bucles

A menudo, en la programación de ordenadores, hay tareas que deben ser realizadas muchas veces. El programa puede contener largas series de secuencias repetitivas de instrucciones para hacer esto, pero con ello no se consigue más que consumir tiempo y espacio. Además, a veces el programador es incapaz de predecir por anticipado cuántas veces debe realizarse una tarea determinada. Por ejemplo, al escribir un

programa de nómina, ¿cuántos cheques debe calcular el programador que serán extendidos? ¿Cinco? ¿Cincuenta? ¿Cuántos empleados tendrá este negocio dentro de seis meses?

En vez de escribir las instrucciones para la misma tarea una y otra vez, el programador prefiere escribir la secuencia una vez, y volver a esa secuencia tantas veces como sea necesario para completar el trabajo. Esa técnica es llamada un *bucle*, y es uno de los conceptos básicos de la programación.

El dBASE utiliza un tipo de bucle llamado bucle-DO. El comando DO inicia una secuencia de instrucciones, que son realizadas repetitivamente hasta que se cumple alguna condición de salida especificada. El bucle empieza con la instrucción DO, que contiene la condición de salida. Termina con un ENDDO. El cuerpo del bucle contiene todas las instrucciones de procesado necesarias para efectuar una sola vez la tarea en particular.

Cuando se inicia el bucle, la secuencia de instrucciones es realizada una primera vez. En algún punto del bucle hay al menos una instrucción que, antes o después, afecta a la condición de salida. ¿Cuáles son esas condiciones de salida? Una muy común es que el valor de alguna variable alcance o exceda un cierto valor:

```
STORE 25 TO SALDO
DO WHILE SALDO > 0
  comandos
ENDDO
```

En algún lugar dentro de ese bucle-DO habrá un comando que reduzca el valor de SALDO, de tal modo que en algún momento se cumplirá la condición de salida.

Si la condición de salida depende de un valor lógico, el bucle termina cuando esa condición se convierte en falsa:

```
STORE T TO MASADLNTE
GO WHILE MASADLNTE
```

o

```
STORE T TO SINERROR
DO WHILE .NOT. SINERROR
```

Si el número de veces que debe ejecutarse el bucle puede ser conocido o calculado, puede empezar el bucle con:

```
DO WHILE STEP# < LIMIT
```

El programa de nómina que veremos más adelante en este mismo capítulo contiene dos bucles-DO distintos, uno para aceptar como dato de input las horas trabajadas por cada empleado, y uno para calcular cada uno de los cheques de nómina.

Bifurcaciones

Los ordenadores no pueden *pensar*. Sin embargo, sí pueden tomar *decisiones*. Lo que hace un ordenador es asimilar una cierta cantidad de texto que usted le especifica, y basándose en lo que ve, proseguir con una de dos o más secuencias de procesado. Hemos visto ya un pequeño ejemplo de esto: el comando DO. Cuando el dBASE llega a un comando DO, examina la condición que ha especificado usted a continuación. Si esta condición es cierta (.NOT. EOF, por ejemplo), el ordenador prosigue con el procesado que se indica tras el comando DO. Si la condición es falsa (*está* en el final del fichero, por ejemplo), el ordenador saltará el procesado que le especifica usted tras el DO y en vez de ello realizará la tarea especificada tras el ENDDO.

Hay otras dos bifurcaciones muy importantes en el dBASE, y una u otra son empleadas casi en cada programa dBASE. Una de ellas es IF, la otra es DO CASE.

IF funciona de una manera muy similar a la forma en que los humanos utilizan esta misma expresión en sus conversaciones. IF (si) tal y tal es cierto, entonces haz eso. IF (si) tal y tal no es cierto, entonces haz ELSE (eso otro). (El uso de la sección ELSE es opcional). He aquí un ejemplo de un comando IF-ELSE generalizado en el dBASE:

```
IF condición
  comandos para la condición cierta
ELSE
  comandos para la condición falsa
ENDIF
```

Cuando el dBASE llega al comando IF, evalúa la condición especificada por usted. Si la condición es cierta, es efectuado el comando que sigue a IF, tras lo cual el programa prosigue más allá del ENDIF. Si la condición es falsa, los comandos de cierto son saltados, y se ejecutan los comandos que siguen a ELSE, tras lo cual el dBASE prosigue más allá del ENDIF.

Los programadores utilizan a menudo algunos medios para proporcionar información a quien lea los listados de los programas. En este caso, observe la indentación en el ejemplo. Éste es un truco habitual para señalar *niveles* allá donde pueden haber varios comandos IF *alojados* juntos, uno dentro del otro. El ELSE y el ENIF asociados con un comando IF en particular son entonces claramente visibles.

Para captar cómo funciona un comando IF, veamos un ejemplo utilizando nuestra base de datos del negocio. Supongamos que el 25 de mayo de 1984 estaba usted calculando cómo organizar las vacaciones de verano, según estas reglas: los empleados que llevaban trabajando menos de seis meses tendrían una semana, los empleados que llevaban trabajando más de seis meses tendrían dos semanas. He aquí una forma en que puede usted programar eso en dBASE con la base de datos de sus empleados:

```
IF ANTIGDAD > 831125
   STORE 1 TO VACACIONES
ELSE
   STORE 2 TO VACACIONES
ENDIF antigdad > 831125
```

Observe la frase «antigdad > 831125» que sigue al ENDIF. No es una parte necesaria de la sintaxis del dBASE; es simplemente una forma por la que el programador sigue el rastro del programa. El dBASE ignora todo lo que haya en la línea después de un ENDIF o un ENDDO, de modo que es un lugar muy conveniente para insertar un comentario dirigido únicamente al programador. Muchos programadores insertan la condición IF o DO con esa forma particular de terminar el END, normalmente en minúsculas para resaltar el texto como un comentario.

El comando CASE es bastante parecido al IF, pero el CASE tiene más posibilidades que las dos elecciones del IF-ELSE. El comando CASE le permite elegir entre varias alternativas. Si ha adoptado usted el indicador de dos dígitos para el mes (01-12), pero desea que el nombre del mes aparezca en los informes, puede tomar el mes (de un cumpleaños, por ejemplo) y trasladar esos dígitos al nombre del mes con un comando CASE. Lo haremos en nuestro informe revisado de cumpleaños. Éste es el formato general:

```
DO CASE
CASE valor="1"
   comandos para el valor=1
CASE valor="2"
   comandos para el valor=2
CASE valor="3"
   comandos para el valor=3
OTHERWISE
   comandos para el caso de no cierto
ENDCASE
```

Cuando el dBASE llega a un comando DO CASE, examina los diferentes CASEs y ejecuta el comando que sigue al CASE que es cierto. Si ningún CASE es cierto, entonces son ejecutados los comandos opcionales OTHERWISE (de otro modo), si figuran. Si ningún CASE es cierto y no hay ningún OTHERWISE, el dBASE sigue simplemente adelante y pasa a la instrucción ENDCASE.

Comprobación de errores

Cuando los programadores preparan sus productos, nunca saben seguro cómo responderá el usuario a ellos. En un punto determinado, el programador espera que sea tecleado un número. ¿Qué ocurre si el usuario teclea en vez de ello caracteres de texto, o un carácter de control? Siempre que sea posible, un programador prudente incluirá en el programa algunas instrucciones extra cuya única finalidad es comprobar cada ítem de datos entrado para averiguar si es el que corresponde.

Observe que no es posible comprobar la exactitud de los datos en sí. La nómina de una persona puede ser cualquier cantidad desde un mínimo absoluto hasta «el techo es el cielo». Podemos, sin embargo, asegurarnos de que los datos sean del tipo correcto: números texto o valores lógicos. El dBASE incluye algunas comprobaciones de error de este tipo. El programador puede especificar, por ejemplo, que solamente sean aceptados dígitos numéricos en un punto particular del programa. En cuanto a esos números, podemos incluso en algunos casos comprobar que su valor se halle dentro de algunos límites razonables. No nos interesa que se extienda ningún cheque semanal por un millón de dólares.

Un aspecto de la comprobación de errores que a menudo no es tomado en consideración es la posibilidad de respuestas inadecuadas cuando todos los datos son correctos. Un ejemplo se produce en nuestro pro

grama de nómina, cuando restamos la cantidad de 13,50$ semanales del seguro de enfermedad al sueldo bruto. Nuestro programa simplificado no tiene en cuenta la posibilidad de un trabajador a tiempo parcial que cobre menos de 13,50$ a la semana; la resta produciría entonces un sueldo neto negativo. Un programa real tiene que tomar en consideración tales eventualidades.

Gran parte de las comprobaciones de error son responsabilidad del programador. Veremos algunas utilizaciones relativamente menores de ello en el programa de nómina al final de este capítulo.

PROGRAMACIÓN DE INFORMES A MEDIDA

Una de las mayores fuerzas del dBASE es su habilidad para preparar informes en cualquier formato que usted desee. La capacidad REPORT, que hemos descrito en el capítulo anterior, es una forma de preparar informes que no necesitan mucha manipulación de datos, o cálculos más allá de sumar varios campos. Ésta no es una limitación básica del dBASE; es simplemente un intento de proporcionar una capacidad sencilla para elaborar algunos informes sin demasiadas complicaciones. Ahora vamos a examinar las capacidades máximas del dBASE para preparar cualquier informe de negocios que desee.

¿Qué tipo de manipulación quiere? Hemos mencionado antes el unir el nombre de pila y el apellido, que por diversas razones se hallan almacenados separadamente. Los distintos cálculos necesarios para cada empleado para calcular el salario bruto, las deducciones, y el salario neto para un informe de nómina, o la impresión directa de los cheques de esa nómina, se hallan más allá de la capacidad del comando REPORT. Si está preparando usted a partir de sus datos un informe donde es necesaria la suma cruzada de hileras y columnas, necesitará un programa especial.

LIMITACIONES DEL EQUIPO

Para examinar el formateo a la medida necesitamos hacer algunas suposiciones respecto a su ordenador. La mayor parte de las pantallas de ordenador tienen veinticuatro hileras de ochenta columnas cada una, y eso es lo que usaremos en nuestros ejemplos. Supondremos también que su display es *direccionable*, es decir, que podemos decirle en qué línea y qué posición de la línea queremos imprimir un carácter.

Hay una gran variedad de impresoras, pero supondremos que su ordenador posee una unidad con un ancho de ochenta columnas y sesenta y seis líneas por página (seis líneas por pulgada, y papel de 8 1/2 por 11 pulgadas).

La mayoría de las pantallas de ordenador permiten ser direccionadas al azar; la línea 10 puede ser escrita después de la línea 20, por ejemplo, aunque vaya antes. Esto nos permite un poco de libertad en el formateo de un informe, cuando el destino es la pantalla. Podemos escribir líneas en cualquier orden de la pantalla. Podemos también retroceder y escribir el principio de una línea *después* de que hayamos escrito su final.

No somos tan afortunados con las impresoras. Muchas impresoras no poseen la capacidad de retroceder, una vez la cabeza impresora ha pasado de una línea determinada, no puede volver atrás. Esto requiere que formateemos cuidadosamente los informes impresos a fin de que toda la impresión de efectúe de principio a fin de la página, y de izquierda a derecha. Si escribe usted programas con output formateado, mi consejo es que utilice siempre un esquema arriba-abajo, izquierda-derecha, aunque en principio planee enviar su informe solamente a la pantalla. Más tarde puede cambiar de opinión y desviar ese informe a la impresora. Evitará un montón de cambios de programación si se prepara para esa posibilidad desde un principio.

Dirigiendo su output

El dBASE está preparado para trabajar tanto con pantalla como con impresora (o con ambas). Seleccionar cuál de las dos quiere es un asunto sencillo, pero existen dos métodos.

Hay veces en que desearemos que el output aparezca tanto en la pantalla como en la impresora. Para ello, dígale al dBASE que SET PRINT ON. Cualquier cosa que sea enviada a la pantalla es enviada también a la impresora. También puede SET CONSOLE OFF para detener en cualquier momento el output a la pantalla. Por supuesto, tiene que tener conectada la pantalla o la impresora si ha de recibir algún output.

Output no formateado

El *formatear* algo significa enviarlo a la pantalla o a la impresora de una forma determinada, con un cuidadoso control sobre lo que se imprime y dónde. Si no nos importa dónde aparece el output en la pantalla o en la página, el dBASE posee dos comandos de output no formateado llamados ? y ??, que podemos utilizar para imprimir nuestra información.

? imprimirá variables, expresiones, literales, o el contenido de un campo sobre la línea siguiente a aquella en que se halle. Por ejemplo, podemos decir:

```
? APELLIDO
```

para imprimir el contenido el campo APELLIDO del registro en uso, ya sea en la pantalla o en la impresora, o en ambos sitios, según hacia donde hayamos dirigido el output del dBASE.

?? hará lo mismo que ?, pero no enviará primero un carácter de salto de línea, como hace el ?. Eso quiere decir que la impresión del ?? se producirá en la misma línea que el anterior output, en el carácter que corresponde a continuación del último. Si tecleamos:

```
? APELLIDO
?? NOM:PILA
```

el apellido y el nombre de pila serán impresos en la misma línea (pero con todos los blancos que se hallen al final del campo APELLIDO entre los dos).

Recuerde que el comando ? simplemente hace avanzar una línea el output, imprimiendo o mostrando una línea debajo de otra (? sin nada a continuación imprime o muestra una línea en blanco). Si usa usted este comando con su impresora, puede añadir instrucciones a su programa para que mantenga un control del número de líneas impresas, a fin de que cada cincuenta líneas o así pueda incluir un comando EJECT para espaciar hasta el principio de la siguiente página. De otro modo, imprimirá sobre las perforaciones de separación de las páginas, si utiliza papel continuo.

El dBASE posee también un método similar de obtener input; de hecho, hay dos comandos para ello. Si los datos del imput son numéricos o lógicos (cierto/falso), la palabra INPUT recibirá esa información del operador. Un uso típico puede ser:

```
INPUT "teclee el número de registros: " TO NUM:REGS
```

Aunque INPUT puede ser utilizado también para cadenas de caracteres tales como los nombres de los empleados, cualquier cadena que sea tecleada debe ir encerrada en apóstrofes o comillas. Una forma más fácil es utilizar la palabra ACCEPT, que no necesita las comillas:

```
ACCEPT "teclee el apellido del empleado: ";
  TO MAPELLIDO
```

Cómo formatear el output

Muy a menudo deseará usted formatear el output del dBASE para controlar qué se imprime y dónde.

Si bien esto es obviamente necesario si está utilizando usted el dBASE para imprimir cheques, donde cada elemento de información debe estar en un lugar determinado, es útil también poseer un formato estándar para cualquier otro informe comercial.

El corazón del formateo a la medida del dBASE es un trío de comandos muy útiles que trabajan juntos: @...SAY...USING. Para utilizar toda la potencia del comando, use una forma distinta del comando SET: SET FORMAT. SET FORMAT TO SCREEN le dirá al dBASE que escriba el output de sus comandos de formateo en la pantalla. Cuando el dBASE empieza a trabajar, la pantalla es el output implícito. SET FORMAT TO PRINT desviará el output a la impresora. Puede usted cambiar libremente de una a otra, siempre que recuerde dónde está en un momento determinado, a fin de que no teclee por ejemplo un comando ERASE a la impresora o un comando EJECT a la pantalla.

@ le permite especificar una localización en cualquier lugar de la pantalla o página impresa en un formato hileras:columnas. Las hileras son numeradas de la 0 a la 23, las columnas de la 0 a la 79. Por ejemplo, la esquina superior izquierda de su pantalla será la localización 0,0, y la esquina superior derecha la 0,79.

Si desea imprimir un mensaje al operador en la esquina superior izquierda, puede teclear:

```
@  0,0 SAY "Hola, operador"
@ 1,14 SAY "¿Cómo te encuentras hoy?"
```

Para asegurarse de limpiar primero la pantalla, teclee ERASE.

SAY imprimirá todo lo que usted le diga, ya sea un literal entre comillas como en el ejemplo de arriba, o el contenido de un campo de una base de datos, o el valor de una variable que haya calculado. Incluso puede utilizar una expresión para el comando SAY: SAY SUELDO*HRAS:SEM imprimirá el producto del sueldo base por el número de horas trabajadas en una semana.

Tiene que mantener usted un cuidadoso control de dónde se encuentra dentro de una línea y lo largo que es lo que tiene que imprimir. Una ayuda para esto es el comando USING, que es una adición opcional al comando SAY. Con él puede especificar un formato particular para el ítem de información que ha de ser impreso. USING espera ver a continuación una frase literal para ser formateada, especificando el número y tipo de caracteres del output. Por ejemplo:

```
USING "9999.9"
```

especifica imprimir un número de cinco dígitos con una cifra decimal. USING acepta seis caracteres de formateo distintos, descritos en la Tabla 6.1. Use un carácter de formateo para cada posición de la impresión. El uso cuidadoso de esos caracteres de formateo puede ahorrarle algo de tiempo de programación y conseguir programas más cortos, como verá en el programa de nómina.

Debe recordar usted que el dBASE no redondea automáticamente los números; en vez de ello, los *trunca*. Esto significa que con una especificación USING "99.9", "49.17" se imprimirá como "49.1" en vez de ser

Tabla 6.1.

Carácter format.	*Significado*
9	sólo un dígito
#	un dígito o los caracteres . + – y espacio
A	sólo un carácter alfabético
X	cualquier carácter
$	imprime un dígito o un signo de dólar en lugar de los ceros guía
*	imprime un dígito o un asterisco en lugar de los ceros guía
!	input de un carácter alfabético, poniendo en mayúsculas las letras minúsculas

correctamente redondeada a "49.2". El redondeo puede conseguirse fácilmente en cualquier programa de ordenador simplemente añadiendo 5 al decimal de la derecha, allá donde quiera redondear el número, inmediatamente antes de la impresión: 49.17 + 0.05 = 49.22, con lo que se imprimirá "49.2" cuando la cifra sea truncada. (Para mantener la exactitud de sus datos, recuerde añadir los 5 de redondeo solamente en el valor de impresión, no en el valor almacenado, que puede ser utilizado para futuros cálculos.)

SAY es sin lugar a dudas una palabra de output; el dBASE de su ordenador interpreta este comando como una instrucción para que envíe alguna información interna a su pantalla o impresora. Hay una palabra equivalente para el input, un comando que el dBASE utiliza para obtener información de usted. Esa palabra es GET, y funciona de la misma manera que SAY. Si teclea:

```
20,10 GET mvariable PICTURE "999"
READ
```

el dBASE aguardará en las coordenadas 20,10 a que usted le teclee un número de uno a tres dígitos. Observe que la palabra PICTURE especifica el formato, del mismo modo que lo hacía USING para una instrucción output.

A menudo encontrará SAY y GET en la misma instrucción. Si está escribiendo un programa que tiene interacción con un operador, es costumbre programar un *prompt*, un mensaje que le diga al operador lo que usted desea justo antes del lugar en el programa donde usted necesita el dato. SAY y GET harán eso. Por ejemplo, su programa puede incluir:

```
@ 20,10 SAY "Entre número empleado":;
  GET MNUM PICTURE "999"
```

Éste es un útil truco de programación para usar en el output de formateo. El comando funcionará tam-

bién utilizando el nombre de una variable en vez de un número específico como localización de una línea o de la posición de un carácter. Cuando haga usted eso, el dBASE buscará el valor de esa variable y lo insertará en el lugar adecuado. Una forma de emplear este útil rasgo es definir dos variables, MCONTL como un contador de línea y MPNTR como un *pointer* o señalizador dentro de una línea. Observe el siguiente grupo de comandos:

```
ERASE
STORE 0 TO MCONTL
STORE 0 TO MPNTR
  @ MCONTL,MPNTR SAY "Hola, operador"
  @ MCONTL+1,MPNTR+14 SAY "¿Cómo te
  encuentras hoy?"
```

Este código es comparable al código más sencillo de arriba, el saludo "Hola" que aún está impreso en 0,0. No obstante, si este método es utilizado al principio de una pantalla, puede usted hacer fácilmente ajustes en toda la pantalla o página arriba y abajo, y centrando a la izquierda o a la derecha, sin cambiar la localización relativa de los datos entre sí. Veamos cómo.

Observe que cada línea después de la primera añade algo a MCONTL para obtener la línea de localización para esa otra línea. En consecuencia, todas las líneas son impresas o mostradas en pantalla *con referencia* a MCONTL en vez de estar *codificadas de forma inamovible* a una línea de localización absoluta. De un modo similar, las posiciones a lo largo de cada línea pueden ser relativas a la localización de principio marcada por MPNTR. Si cambia usted el valor inicial de MCONTL de 0 a 4, dejará cuatro líneas blancas en la parte de arriba de la pantalla. Todas las demás líneas en que se use MCONTL como indicador de línea se moverán correspondientemente hacia abajo. Del mismo modo, cambiar el valor inicial de MPNTR de 0 a 10 moverá toda la pantalla 10 espacios hacia la derecha, ajustando el margen izquierdo.

Si utiliza usted números específicos de localización para el número de línea y la posición del carácter en cada comando de una larga pantalla de información, un cambio del espaciado arriba/abajo o izquierda/derecha implicará un cambio para cada comando en esa pantalla, quizá más de dos docenas.

CONSTRUYENDO UN FICHERO DE FORMATOS

Supongamos que utiliza usted el mismo formato en más de un lugar. Los cheques, por ejemplo, pueden ser formateados del mismo modo tanto si son cheques de nómina como cheques a proveedores. En consecuencia, necesitará los mismos comandos de formateo en dos o más lugares.

La capacidad de formateo del dBASE le permite construir un fichero .FMT de comandos de formateo, del mismo modo que el comando REPORT construye un fichero de formatos del tipo .FRM. Supongamos que disponemos de un fichero de formato de cheques llamado CHEQUES.FMT. Puede utilizar ese fichero incluyendo:

```
SET FORMAT TO CHEQUES
```

en cualquier programa en el que necesite preparar cheques.

UN PRIMER PROGRAMA

Para nuestro primer programa queremos algo sencillo. En el último capítulo mostramos cómo imprimir un listado de los cumpleaños de los empleados usando el comando REPORT. Ahora redactaremos un programa sencillo para imprimir un listado de cumpleaños con las fechas dispuestas de un modo más orientado a nuestros ojos.

En primer lugar, definamos el output que quere-

mos. Queremos un listado impreso de cumpleaños con la fecha en primer lugar, en el formato típico, como 25 de mayo de 1984. Luego queremos listar el nombre de pila y el apellido con un solo espacio entre los dos. Ya que estamos en ello, el propietario de la compañía tal vez desee llamar personalmente a cada empleado para desearle feliz cumpleaños, así que podemos añadirle su extensión telefónica.

Este definir primero cómo queremos que sea presentado el output es uno de los pasos principales en escribir un programa dBASE. No se lo salte nunca, o le traerá más trabajo cuando empiece a codificar realmente el programa. Empiece siempre sus esfuerzos de programación con unas bien definidas especificaciones de output.

El dBASE contiene un editor que le permite escribir su fichero de programa desde dentro del propio dBASE. Usted crea un fichero de programa en dBASE tecleando:

```
MODIFY COMMAND «nombrefichero»
```

El dBASE abrirá un fichero llamado *nombrefichero*.CMD (*nombrefichero*.PRG en los ordenadores de 16 bits). Luego simplemente teclee los comandos que desea ejecutar, siguiendo cada uno de ellos con un retorno del carro o la tecla ENTER. Cuando haya terminado con el fichero y desee almacenarlo, teclee control-W (^W), y el dBASE salvará el fichero en su disco. Luego puede ejecutar ese programa tecleando:

```
DO «nombrefichero»
```

También puede utilizar los más populares programas de tratamiento de textos para preparar ficheros de comandos, si se siente más cómodo con ellos. Algunos programas de tratamiento de textos, como el WordStar, contienen rasgos que no están en el editor del dBASE. Pueden hacerle un poco más fácil el escribir un programa. Por ejemplo, una función de *búsqueda* y

sustitución global es muy útil si desea cambiar el nombre de una variable que aparece en muchos lugares de su fichero. Puede teclearla una vez y dar instrucciones a su programa de tratamiento de textos para que la remplace allá donde aparezca en su nuevo texto. Asegúrese de todos modos de nombrar su fichero utilizando el tipofichero .CMD o .PRG, que son los correspondientes a su versión del dBASE.

El programa definitivo del listado de cumpleaños, LISTCUMP, aparece como Listado 6.1. Lo examinaremos línea a línea.

Cualquier línea en un programa dBASE que empieza con un asterisco es una línea de comentario, que el dBASE ignora. Los programadores utilizan esos comentarios para varias finalidades: títulos, o documentación interna de lo que está ocurriendo en el programa en ese punto en particular. Dejar comentarios en un programa hace que el programa se ejecute un poco más lento, por lo que a veces son eliminados antes de que el programador lo entregue al usuario. Los programas muy comentados son una de las mejores formas de describir y documentar un programa, pero aquí no tenemos espacio para hacerlo así. Hemos incluido solamente el título y la descripción del programa en las primeras tres líneas.

CLEAR sitúa el dBASE en una condición conocida: borra las variables de la memoria, selecciona la pantalla como dispositivo de output, y establece (SET) los parámetros en sus valores implícitos. Primero queremos decirle al dBASE que no nos informe de lo que está haciendo paso tras paso, así que lo acallamos con SET TALK OFF, luego cambiamos el output a la impresora con SET FORMAT TO PRINT. Los comandos STORE establecen valores iniciales para nuestro contador de líneas y señalizador de caracteres, y carga nuestra cabecera en la variable MSJ.

El primer comando es un tanto inusual. Lo que queremos hacer es imprimir nuestro título en el centro de una página. Puesto que hemos supuesto una impresora de 80 columnas, el centro se halla en la columna

Listado 6.1.

```
* PROGRAMA LISTCUMP
* este programa imprime una lista de los cumpleaños
* de los empleados, trasladando los datos a forma normal
CLEAR
SET TALK OFF
STORE 0 TO MCONTL
STORE 10 TO MPNTR
STORE "LISTA DE CUMPLEAÑOS DE LOS EMPLEADOS";
   TO MSJ
 MCONTL, (40–LEN(MSJ)/2) SAY MSJ
STORE MCONTL+2 TO MCONTL
USE PERSONAL INDEX LSTCMPL
DO WHILE .NOT. EOF
  STORE $(FCHA:NAC,1,2) TO MES
  DO CASE
   CASE MES = "01"
    STORE $(FCHA:NAC,3,2) + " DE ENERO" TO MSJ
   CASE MES = "02"
    STORE $(FCHA:NAC,3,2) + " DE FEBRERO" TO MSJ
   CASE MES = "03"
    STORE $(FCHA:NAC,3,2) + " DE MARZO" TO MSJ
   CASE MES = "04"
    STORE $(FCHA:NAC,3,2) + " DE ABRIL" TO MSJ
   CASE MES = "05"
    STORE $(FCHA:NAC,3,2) + " DE MAYO" TO MSJ
   CASE MES = "06"
    STORE $(FCHA:NAC,3,2) + " DE JUNIO" TO MSJ
   CASE MES = "07"
    STORE $(FCHA:NAC,3,2) + " DE JULIO" TO MSJ
   CASE MES = "08"
    STORE $(FCHA:NAC,3,2) + " DE AGOSTO" TO MSJ
   CASE MES = "09"
    STORE $(FCHA:NAC,3,2) + " DE SETIEMBRE" TO MSJ
```

```
    CASE MES = "10"
     STORE $(FCHA:NAC,3,2) + " DE OCTUBRE" TO MSJ
    CASE MES = "11"
     STORE $(FCHA:NAC,3,2) + " DE NOVIEMBRE" TO MSJ
    CASE MES = "12"
     STORE $(FCHA:NAC,3,2) + " DE DICIEMBRE" TO MSJ
  OTHERWISE
    STORE "NO ESPECIFICADO" TO MSJ
  ENDCASE NOT EOF
  @ MCONTL,MPNTR SAY MSJ
  @ MCONTL,MPNTR+15 SAY TRIM(APELLIDO) + ", ";
     + TRIM(NOM:PILA)
  @ MCONTL,MPNTR+41 SAY EXTN:TEL USING "XXXX"
  STORE MCONTL+2 TO MCONTL
  SKIP
ENDDO no eof
USE
EJECT
SET FORMAT TO SCREEN
SET TALK ON
RELEASE ALL
RETURN
```

---

40. Para centrar el título, nos movemos hacia el borde de la izquierda la mitad de la longitud del mismo. Nuestro *pointer* de la columna, en consecuencia, deberá estar a:

```
(40 - LEN(MSG)/2)
```

lo cual significa 40 columnas menos la mitad de la longitud del contenido de MSG. Ésta es una forma general de escribir algo centrado automáticamente en la impresora o en la pantalla.

El primer bucle-DO examina el fichero PERSONAL para ver si la posición actual es un final de fichero. Si

no (.NOT. EOF es cierto), son ejecutados los comandos de dentro del bucle-DO. El primero de ellos utiliza el *operador de subcadenas*, $. $ es un comando dBASE versátil, que permite tomar uno o más caracteres de dentro de un grupo más grande de caracteres llamado *cadena*. Aquí, $(FCHA:NAC,1,2) toma del campo de la fecha de nacimiento de un empleado los dos caracteres que empiezan en la posición 1. Esos caraceres representan el mes, y son almacenados en una variable llamada MES. (Vea el programa NOMINA para una utilización ligeramente distinta del comando $.)

Luego el dBASE ejecuta nuestro DO CASE, que consiste en trece comandos, del que sólo uno es ejecutado. Si los dos caracteres que hemos almacenado en MES representan alguno de los doce meses 01-12. entonces el nombre del mes correspondiente será almacenado en MSG con un solo blanco tras él. Utilizando de nuevo el operador de extracción $, podemos extraer el tercer y cuarto caracteres de FCHA:NAC (el día del mes) y añadirlo al nombre para formar una fecha completa. Puesto que nos gusta prever cualquier contingencia no prevista, añadimos a los doce comandos de los meses un OTHERWISE, que detectará los registros con datos ausentes o erróneos.

Observe que hemos utilizado indentaciones para hacer el fichero del programa más legible. Los espacios marcados al principio de las líneas son ignorados por el dBASE, pero son una clara ayuda a las personas que leen el programa. También debemos mencionar que el dBASE no lee nada de una línea END después del comando END en sí, de modo que es usted libre de situar sus comentarios en esa línea sin ningún asterisco. Es una buena práctica identificar el bucle en particular que termina ahí con una repetición en minúsculas de la condición del bucle. Esto funciona también como una comprobación de que ha terminado usted todos los bucles con un comando END. Puede ver ejemplos de ello en el programa LISTCUMP.

Tras trasladar el mes, que como puede ver ocup casi la mitad del programa, imprimimos el contenid

de MSJ (mes y fecha). Limpiamos los nombres de los empleados mejor de lo que lo hace REPORT con TRIM, que elimina los blancos finales del apellido, añadiendo una coma y un solo blanco, y luego añadiendo el nombre de pila. A un espacio conveniente tras el nombre, imprimimos la extensión telefónica, y luego nos movemos dos líneas hacia abajo añadiendo dos a nuestro contador de líneas.

SKIP nos traslada al siguiente registro, que será o bien el registro de un empleado en activo o el marcador de final de fichero. Cuando ENDDO nos lleve de vuelta a la condición DO, avanzaremos de nuevo a lo largo de todas las instrucciones para el siguiente empleado, o bien dejaremos el bucle y seguiremos después del ENDDO si estamos al final del fichero.

Terminamos con EJECT la página final de la impresora, luego volvemos de nuevo a la pantalla, le decimos al dBASE que vuelva a hablarnos, borramos las variables, y terminamos el programa con RETURN. Si lo deseamos, podemos almacenar más texto o comentarios tras el RETURN sin el *, porque el dBASE deja de actuar tras el RETURN.

El programa producirá un informe como el que se refleja en el Report 6.1.

UN PROGRAMA DE NÓMINA

Ahora que hemos vagabundeado un poco por los perímetros de la programación en dBASE para entender algunos aspectos fundamentales, ya es tiempo de iniciar nuestro programa de nómina. Lo que haremos primero es preparar un programa conductor, NOMINA, que recorra la base de datos para seleccionar a los empleados a quienes hay que preparar cheques. Luego prepararemos el programa subsidiario, CHEQUENOM, para calcular e imprimir el cheque de uno de los empleados.

Por favor, no olvide que, dentro de las características de este manual, solamente podemos efectuar una

Report 6.1.

LISTA DE CUMPLEAÑOS DE LOS EMPLEADOS

18 DE ENERO	PITTSINGER, ARNOLD	117
27 DE ENERO	MOZART, WILLIE	110
10 DE FEBRERO	OSVALDO, MARIANNE	112
23 DE FEBRERO	HANDLE, GEORGE	103
19 DE MARZO	ROSEKRANZ, THELMA	101
21 DE MARZO	BOCK, JOHN	105
31 DE MARZO	HIDEN, JOE	107
21 DE ABRIL	JOHNSON, JIMMY	109
07 DE MAYO	BRAMZ, JOHN	108
30 DE MAYO	SVENSON, ARNE	113
04 DE JULIO	FUSBANCK, RENATE	105
28 DE JULIO	LORE, HANNAH	114
30 DE SETIEMBRE	OCHOA, JUAN	117
17 DE OCTUBRE	OETTINGER, ANNAMARIE	106
27 DE NOVIEMBRE	RANSOM, EMMA	102
10 DE DICIEMBRE	FRANCK, CEASAR	115
16 DE DICIEMBRE	RANSOM, ROGER	100
16 DE DICIEMBRE	BAYTOVEN, LOU	114

breve introducción a la programación dBASE. Como medio de programación, el dBASE es mucho más potente que algunos de los más antiguos lenguajes de programación. Virtualmente puede hacer cualquier tarea con datos que usted le pida.

Pensemos por un momento cómo hacer una nómina. La primera y más importante actividad es obtener la información de las fichas de registro de entradas y salidas, las horas que cada empleado trabajó la semana pasada. Luego, para cada empleado, tenemos que efectuar los cálculos que den como resultado el importe del cheque: multiplicar las horas del empleado por el salario base aplicable, añadir los pluses de horas extras (las que pasen de las cuarenta semanales), luego restar los impuestos y los pagos por seguros para llegar al salario neto. Dispondremos un formato determinado para los cheques, y prepararemos en la impresora los cheques en sí y un informe de la nómina.

Nómina

Incluiremos en esta sección un programa de nómina a fin de que pueda empezar a ver cómo pueden ser manejados los problemas más complejos por el dBASE. Para encajar el programa a las proporciones de este manual hemos omitido algunas de las subrutinas, como el redondeo de los números calculados.

Puesto que los cálculos de las retenciones federales y estatales y los pagos de seguros pueden variar mucho de compañía a compañía y de estado a estado, nuestro programa NOMINA supondrá una retención federal de un 15 % y una retención estatal de un 5 %, y un coste fijo semanal del seguro de 13,50$ por empleado. Supondremos también que los propietarios del negocio, los Ransom, no entran en la nómina semanal.

El listado 6.2 muestra nuestro programa NOMINA maestro. Espero que a estas alturas se vea usted capaz de seguir el programa para ver cómo funciona. El programa NOMINA se ve muy simplificado efectuando buena parte del trabajo repetitivo en *subrutinas*, otros ficheros de programas dentro del programa NOMINA. Esas subrutinas, FCHASREG y CHEQUENOM, son ejecutadas cuando son llamadas cada vez por DO *nombre subrutina* dentro del bucle, una vez para cada empleado. Cuando DO es utilizado de este modo para llamar a otro programa, no debe confundirse con un bucle-DO. Un comando-DO que inicia un bucle debe ser equilibrado con un comando END, de modo que el principio y el fin del bucle queden definidos. Puesto que el fin de una subrutina queda definido por una instrucción RETURN en la propia subrutina, DO *nombre subrutina* no necesita ningún comando END correspondiente.

Necesitamos una explicación rápida de un nuevo uso del operador de subcadenas $. En el listado del programa de los cumpleaños, utilizamos $ para extraer una subcadena en la que estábamos interesados, los caracteres que representaban el mes. En el programa de nómina, utilizamos $ para comprobar si Ransom figura en el apellido. “RANSOM”$(APELLI-

DO) devolverá un valor de cierto o falso; si ponemos .NOT. delante invertiremos ese valor, asi que .NOT. "RANSOM"$(APELLIDO) es cierto para cualquier empleado excepto los Ransom.

Listado 6.2.

```
* PRG. NOMINA
* programa maestro de nómina de empleados
* las variables son nivel A
CLEAR
SET TALK OFF
SET FORMAT TO SCREEN
* bucle #1 para obtener inf de fchasreg
USE PERSONAL
DO WHILE .NOT. EOF
  IF .NOT. "RANSOM"$(APELLIDO)
    DO FCHASREG
  ENDIF not ransom
  SKIP
ENDDO not eof 1
ERASE
* bucle #2 para calcular chequenom
GOTO TOP
DO WHILE .NOT. EOF
  IF .NOT. "RANSOM"&(APELLIDO)
    DO CHEQUENOM
  ENDIF not ransom
  SKIP
ENDDO not eof 2
GOTO TOP
DO REPRTNOM
USE
CLEAR
SET TALK ON
RETURN
* fin del programa
```

Listado 6.3.

```
*PRG. FCHASREG
* entra la información de las fichas registro
* para un empleado las variables son nivel B
STORE T TO BHRS:NOTOK, BHRS:OK
DO WHILE BHRS:NOTOK
  ERASE
  STORE 0 TO BHORAS, BLCONT, BPNTR
    @BLCONT+10,BPNTR+20 SAY "registre horas
       trabajadas la última semana:"
  STORE TRIM(APELLIDO) + ", " + TRIM(NOM:PILA) ;
     + " " TO BMSJ
    @BLCONT+12,BPNTR+20 SAY BMSJ GET ;
       BHORAS PICTURE "99.9"
  READ
  IF BHORAS >= 0
      @BLCONT+14,BPNTR+20 SAY BHORAS ;
         USING "99.9"
      @BLCONT+14,BPTNR+25 SAY "HORAS REGIS;
         TRADAS. ¿ES CORRECTO? (S/N)"
      @BLCONT+14,BPTNR+64 GET BHRS:OK
    READ
    IF BHRS:OK
      REPLACE HRASSEM WITH BHORAS
      STORE F TO BHRAS:NOTOK
    ENDIF bhrs:ok
  ELSE
      BLCONT,BPNTR SAY "NUMEROS NEGATIVOS ;
         NO PERMITIDOS"
    ENDIF bhoras >=0
 ENDDO bhrs:notok
 RELEASE ALL LIKE B*
 RETURN
 *fin subrutina
```

Listado 6.4.

```
* PRG. CHEQUENOM
* calcula e imprime el cheque de un empleado
* las variables son nivel C
REPLACE BRUT:SEM WITH SUELDO*HRASSEM
IF HRASSEM > 0
  IF HRASSEM > 40
    REPLACE BRUT:SEM WITH BRUT:SEM + ;
      ((HRASSEM-40)*SUELDO/2)
  ENDIF hrassem > 40
  REPLACE BRUT:ANL WITH BRUT:ANL + BRUT:SEM
  REPLACE IMPFED:SEM WITH BRUT:SEM*0.15
  REPLACE IMPFED:ANL WITH IMPFED:ANL + ;
    IMPFED:SEM
  REPLACE IMPEST:SEM WITH BRUT:SEM*0.05
  REPLACE IMPEST:ANL WITH IMPEST:ANL + ;
    IMPEST:SEM
  REPLACE SEG:ANL WITH SEG:ANL + 13.50
  REPLACE NET:SEM WITH BRUT:SEM - IMPFED:SEM ;
    - IMPEST:SEM - 13.50
  REPLACE NET:ANL WITH NET:ANL + NET:SEM
  * imprimir un cheque en formato continuo
  SET FORMAT TO PRINT
  @ 5,70 SAY DATE( )
  @ 8,20 SAY TRIM(NOM:PILA) + " " + ;
    TRIM(APELLIDO)
  @ 8,50 SAY NET:SEM USING "$$$$$$.99"
  EJECT
  SET FORMAT TO SCREEN
ENDIF hrassem > 0
RELEASE ALL LIKE C*
RETURN
* fin de la subrutina
```

Listado 6.5.

```
* PRG. REPRTNOM
* imprime informe de la nómina
* las variables son nivel D
SET TALK OFF
ERASE
* asegúrese de que el papel está OK
STORE T TO DNOPAPEL
STORE " " TO DANSR
DO WHILE DNOPAPEL
  IF DANSR <> " "
    @ 9,27 SAY DANSR USING "A"
    @ 9,29 SAY "NO ES UNA RESPUESTA CORRECTA."
  ENDIF dansr <> " "
  @ 11,23 SAY "PREPARE IMPRESORA PARA INFORME
     NOMINA."
  @ 13,19 SAY "SAQUE CHEQUES, INTRODUZCA ;
     PAPEL A PRINCIPIO PAGINA."
  @ 15,24 SAY "pulse Y si está listo para ;
     imprimir." GET DANSR PICTURE "!"
  READ
  IF DANSR = "Y"
  STORE F TO DNOPAPEL
  ENDIF dansr=Y
ENDDO dnopapel
ERASE
SET FORMAT TO PRINT
STORE 0 TO DLCONT,DPNTR
  DLCONT,DPNTR SAY "REPORT NOMINA PARA "
  DLCONT,DPNTR+19 SAY DATE( )
STORE DLCONT+2 TO DLCONT
  DLCONT,DPNTR SAY "Empleado"
  DLCONT,DPNTR+31 SAY "Bruto"
  DLCONT,DPNTR+40 SAY "ImpFed"
```

```
  DLCONT,DPNTR+48 SAY "ImpEst"
  DLCONT,DPNTR+56 SAY "Seg"
  DLCONT,DPNTR+64 SAY "Salr Net"
STORE DLCONT+2 TO DLCONT
* bucle recorriendo la base de datos
DO WHILE .NOT. EOF
  IF .NOT. "RANSOM"$(APELLIDO) .AND. HRASSEM > 0
    STORE TRIM(APELLIDO) + ", " + NOM:PILA ;
       TO DMSJ
      DLCONT,DPNTR SAY DMSJ
      DLCONT,DPNTR+30 SAY BRUT:SEM USING ;
         "9999.99"
      DLCONT,DPNTR+40 SAY IMPFED:SEM USING ;
         "999.99"
      DLCONT,DPNTR+48 SAY IMPEST:SEM USING ;
         "99.99"
      DLCONT,DPNTR+56 SAY "13.50"
      DLCONT,DPNTR+4 SAY NET:SEM USING ;
         "9999.99"
    STORE DLCONT+1 TO DLCONT
  ENDIF not ransom y hrassem > 0
  SKIP
ENDDO not eof
EJECT
SET FORMAT TO SCREEN
SET TALK ON
RELEASE ALL LIKE D
RETURN
* fin de la subrutina
```

---

7. PROGRAMACIÓN FÁCIL EN dBASE

En los primeros capítulos de este manual vimos algunos de los comandos que el dBASE acepta para manipular y mostrar datos. En el capítulo 6, vimos cómo acumular esos comandos en ficheros llamados *programas*. El dBASE no es solamente un gestor de base de datos, es también un lenguaje de programación de moderada potencia.

Pero enfrentémonos a ello: la programación no es para todo el mundo. Para algunos, es un ejercicio de excesivo detalle y frustración. Hallar un error (o *bug*) es un problema que puede convertirse en una experiencia más bien desagradable para algunas personas.

¿Tiene que escribir usted sus propios programas si utiliza el dBASE? No, en absoluto. Dispone de otras varias opciones. En primer lugar, puede trabajar con el dBASE únicamente en modo interactivo. Aunque esto le impide utilizar muchos de los versátiles rasgos del dBASE, evita el problema de la programación.

Segundo, puede comprar programas escritos en dBASE que harán el trabajo por usted. Pero, tratándose de programas de utilidad general, raramente se presentan en el formato exacto que usted desea. Ayuda mucho el ser capaz, como mínimo, de modificar el programa que ha comprado para que realice el trabajo que usted desea.

Para las personas que desean evitar la mayor parte de la programación detallada, hay algunos caminos

más fáciles. Debido al éxito del dBASE en el mercado, Ashton-Tate (el fabricante del dBASE) y algunas otras compañías han ofrecido ayudas de programación para que el no programador utilice los datos del dBASE. Usándolas, puede usted realizar su trabajo sin tener que preocuparse de interioridades y detalles. En este capítulo consideraremos algunos de esos productos. Observe que, a excepción del ZIP, no forman parte del paquete dBASE que ha comprado usted. Tendrá que obtenerlas separadamente.

ALGUNAS AYUDAS PARA LA PROGRAMACIÓN dBASE

FRIDAY!

El *Robinson Crusoe* de Daniel DeFoe acuñó una nueva frase en inglés: el «hombre Friday» o la «mujer Friday» señala a alguien capaz de ayudar en cualquier cosa que se presente. Ashton Tate, que ofrece el dBASE, ofrece también un producto, Friday!, que será de considerable ayuda a muchos usuarios de las bases de datos.

En primer lugar, olvide tener que aprender cómo usar los comandos dBASE. Friday! es un programa interactivo que genera ficheros de bases de datos para usted y le permite que los manipule tanto como sea necesario. Todo lo que tiene que hacer usted es responder a las preguntas que Friday! le plantee. Por supuesto, bajo el enunciado de «no hay ninguna comida gratis», hay dos penalizaciones a esta facilidad de uso. Una es un tiempo más lento de ejecución. Aunque Friday! ejecutará sus tareas, lo hará un poco más lento de lo que haría el mismo trabajo un fichero programado a la medida. La segunda penalización es una versatilidad ligeramente disminuida. Friday! no posee la totalidad de las capacidades contenidas en el dBASE.

Friday! no es un programa usado *con* el dBASE; es un producto independiente que contiene muchos de

los rasgos del dBASE. El fichero de datos que genera es utilizable también con el dBASE.

Friday! opera de una forma realmente amistosa. Presenta una serie de *menús*: un grupo de actividades que puede ejecutar a cada paso de su operación. Usted selecciona lo que desea que haga de entre las posibilidades de un menú dado. Cuando Friday! necesita información de usted, le formula una pregunta determinada y aguarda su respuesta.

ZIP

Gran parte de la interacción del dBASE con el operador implica algún tipo de formateo. Esto puede ser una pantalla llena de información que el ordenador desea que vea el operador, o una pantalla de prompts para que el operador teclee allá sus datos, o un formato lleno de datos para la impresora. En el capítulo anterior vimos la serie de comandos @...SAY...GET, extremadamente útiles en preparar pantallas y formatos.

Puede ser muy tedioso y lento calcular y escribir todos los distintos comandos que requiere generar una pantalla así de formatos. ¿No sería útil conseguir que el ordenador lo hiciera por usted?

Con el dBASE, Ashton-Tate proporciona un producto auxiliar denominado ZIP. Con ZIP, usted simplemente teclea su pantalla o formato exactamente tal como quiere que aparezca, y ZIP prepara la secuencia de comandos dBASE necesaria para crear esa pantalla o formato. ZIP preparará incluso el código dBASE de un fichero de formateo que haya generado usted con su programa de tratamiento de textos.

Supongamos que desea preparar una pantalla para mostrar un formato particular de factura. Utilizando ZIP, usted simplemente tecleará en la pantalla la información que desee presentar. Allá donde quiera un prompt o identificador, tecléelo. Allá donde desee un dato de información de la factura almacenada en la base de datos, teclee en la pantalla el carácter y el

nombre de la variable que contiene el dato: el cliente le indicará a ZIP que en ese lugar desea usted que se imprima el nombre del cliente. En cualquier punto que desee que el operador teclee algún dato, muestre en la pantalla el símbolo # seguido por la variable en la cual se halla ese dato: #numcliente significará que ZIP debe esperar a que el operador teclee el número del cliente en ese punto.

Cuando usted haya terminado, ZIP empezará su trabajo. Generará un fichero .CMD (o .PRG) de comandos dBASE. Para cada posición de la pantalla donde hay que imprimir información, contará filas y columnas y preparará una instrucción con las coordenadas correspondientes y el texto que usted desea ahí. Para cada instrucción GET donde hay que introducir información, ZIP calculará las coordenadas, preparará la porción y el prompt SAY, si es necesario, y la variable GET. Sin embargo, no utiliza los comandos PICTURE y USING.

Utilice o no alguna de las otras ayudas de programación dBASE, descubrirá que ZIP es un auxiliar extremadamente útil. Se halla incluido en su sistema dBASE de Ashton-Tate.

dPROGRAMMER

Una reciente adición al grupo de programas auxiliares del dBASE es un generador de programas llamado dPROGRAMMER. Un generador de programas es un programa de ordenador que, bajo la guía del operador, genera otros programas de ordenador.

El dPROGRAMMER es un sistema *dirigido por menús*; presenta una serie de menús para que el operador seleccione la actividad. Incluye instrucciones *on-line*, es decir que puede hallar usted apoyo directamente del ordenador en vez de tener que acudir constantemente al manual. El dPROGRAMMER contiene también rasgos específicos que facilitan la utilidad del dBASE para la contabilidad personal o de pequeños negocios.

QUICKCODE

Tras la introducción del dBASE II, una de las primeras compañías en ofrecer programas auxiliares fue Fox & Geller en Nueva Jersey. Sacaron al mercado el QUICKCODE, un generador de programas que trabaja con el dBASE II.

En el último capítulo examinamos cómo escribir un programa dBASE. Aquellos que no deseen escribir programas pueden utilizar el QUICKCODE para generar algunas de las menos complicadas aplicaciones programadas.

QUICKCODE empieza suponiendo que usted utilizará la pantalla para reproducir en su beneficio los datos de entrada que teclee. Así, el primer paso es definir el aspecto que quiere que tenga esa pantalla mientras entra los datos. Al mismo tiempo, define usted cómo quiere que aparezcan los datos en su base de datos. Luego define el formato del output. Ambas pantallas son definidas creándolas realmente sobre la pantalla en blanco de su ordenador.

Los programas dBASE interactúan con el operador presentando *menús* de actividades o elecciones que el operador puede seleccionar. QUICKCODE le permite crear muy rápidamente sus propios menús, con la capacidad QUICKMENU. Especialmente valiosa es la habilidad del QUICKCODE de preparar ficheros de su base de datos para usarlos con el WordStar, un popular programa de tratamiento de textos, y el MailMerge, el programa auxiliar del WordStar de listado de direcciones.

Existe, por supuesto, una penalización por usar los generadores de programas. Los programas que producen raras veces se ejecutarán tan rápido o eficientemente como uno cuidadosamente programado a la medida. Para aquellos que no desean programar, sin embargo, éste puede ser un precio pequeño que estén dispuestos a pagar.

ACCESORIOS DE PROGRAMACIÓN dBASE

Si decide probar usted la programación dBASE, hay unas cuantas herramientas en el merado en las que puede que esté interesado. Menciono más adelante algunas de ellas, describiendo sus capacidades más típicas. Cada una de ellas, sin embargo, pueden proporcionar una ayuda adicional que se halla más allá de lo que puede ser descrito aquí.

Cuando escribe usted un programa, desea documentar su operación muy concienzudamente al menos por dos razones. La primera, que si el programa funciona incorrectamente, deseará ser capaz de retroceder a cada paso y recordar cuál es ese paso. La segunda, que una vez haya experimentado la utilidad que le proporciona el dBASE, probablemente deseará modificar sus programas para incrementar aún más su capacidad. He mencionado antes que incluir comentarios dentro de un programa es un poderoso medio de documentar lo que está haciendo. El problema es que incluir esos comentarios resta un poco de tiempo al dBASE cuando ejecuta ese programa, mientras el dBASE decide que esa línea es un comentario, no una orden, y la salta para buscar el siguiente comando. Ayuda mucho el escribir el programa completamente documentado, y luego efectuar una copia operativa de la que se hayan borrado todos los comentarios.

También ayuda, como he señalado en los capítulos dedicados a la programación, indentar los códigos de su programa para ver los distintos niveles a la primera ojeada. Un programa adecuadamente llamado PRETTY hace esas dos tareas por usted. PRETTY es un programa auxiliar por el cual pude pasar usted su programa codificado. PRETTY, si usted le da las instrucciones al efecto, extraerá todos los comentarios e indentará los distintos niveles. Su uso es muy sencillo: teclee:

```
PRETTY fuente.CMD destino.CMD $I2BCM8
```

y su programa fuente.CMD será limpiado y almacenado en cualquier fichero que haya indicado usted como destino.CMD. Los datos después del signo del dólar representan opciones que puede usted cambiar. I2 indica indentar dos espacios por nivel, B significa quitar líneas en blanco, C requiere la anulación de las líneas de comentarios, y M8 da instrucciones a PRETTY de que deje un margen de ocho espacios a la izquierda (para taladrar la hoja y archivarla, por ejemplo).

dUTIL es otro programa que hace cosas útiles con su fichero de programa. Un estupendo rasgo del dUTIL, además de indentar, es que situará los comentarios correspondientes tras sus comandos ENDDO y ENDIF, un rasgo útil de documentación que ya he descrito antes. Si una estructura IF-ENDIF o DO-ENDDO no está convenientemente cerrada, el dUTIL le ayudará a encontrarla. Si ha escrito usted todo su programa en minúsculas, el dUTIL lo recorrerá y pondrá en mayúsculas todas las palabras clave dBASE. En un programa impreso después de que el dUTIL haya terminado su trabajo, los comentarios y los comandos dBASE son fácilmente distinguibles.

Otro programa de utilidad en el que puede estar usted interesado es el dCREF. Su utilidad es empleada principalmente para generar listados impresos de sus programas. Entre otras cosas, pagina sus ficheros de comandos en claros bloques. También crea e imprime una tabla de referencias cruzadas para mostrar cada variable de memoria, nombre de campo y nombre de archivo referidos en un programa, y dónde aparecen esos nombres dentro del programa. Esto es extremadamente útil si está buscando un *bug* (error) o cambiando uno de los nombres.

MÓDULO RUN-TIME

Si cree que puede vender los programas dBASE que ha escrito, Ashton-Tate tiene algo inusual para usted. Normalmente, quien utilice su programa nece-

sitará poseer un dBASE para ejecutar cualquier código dBASE. Sin embargo, con el Módulo Run-Time de Ashton-Tate puede vender usted su programa a alguien que no posea un dBASE II. El Módulo Run-Time duplica básicamente la porción de ejecución del dBASE. Puede usted venderla junto con su programa dBASE.

Pero hay un beneficio adicional en el Módulo Run-Time. Normalmente, cuando vende usted un programa dBASE, tiene que vender el *código fuente*, el texto en inglés para el programa. Con el Módulo Run-Time, usted recibe un programa codificador, que toma su código fuente y produce un fichero de programa que no es texto. Este fichero codificado es el que vende usted con el Módulo Run-Time. En consecuencia, nadie puede obtener su código fuente.

El Módulo Run-Time se halla disponible para programadores bajo licencia de Ashton-Tate.

APÉNDICE: EJEMPLO DE BASE DE DATOS

Nota: Los campos a partir de HRAS:SEM cambiarán cada vez que sea ejecutado el programa; por ello no aparecen en los REGISTROS #00002 a #00018.

```
REGISTRO # 00001
     NUM:EMPL        :0001:
     ANTIGDAD        :801001:
     EXTN:TEL        : 100:
     NOM:PILA        :ROGER      :
     APELLIDO        :RANSOM          :
     CALLE           :1234 ELM STREET      :
     CIUDAD          :COVINA        :
     ESTADO          :CA:
     COD:POSTAL      :91723:
     TEL:PART        :2135551234 :
     NO:SEG:SOC      :           :
     EXENC           : 4:
     FCHA:NAC        :1216:
     SUELDO          :20.540:
     HRAS:SEM        : 0.0:
     BRUT:SEM        :    0.00:
     BRUT:ANL        :      0.00:
     IMPFED:SEM      :  0.00:
     IMPFED:ANL      :    0.00:
     IMPEST:SEM      : 0.00:
     IMPEST:ANL      :    0.00:
     SEG:ANL         :    0.00:
     NET:SEM         :    0.00:
```

```
REGISTRO # 00002
     NUM:EMPL       :0002:
     ANTIGDAD       :801001:
     EXTN:TEL       : 102:
     NOM:PILA       :EMMA       :
     APELLIDO       :RANSOM          :
     CALLE          :1234 ELM STREET      :
     CIUDAD         :COVINA         :
     ESTADO         :CA:
     COD:POSTAL     :91723:
     TEL:PART       :2135551234 :
     NO:SEG:SOC     :           :
     EXENC          : 4:
     FCHA:NAC       :1127:
     SUELDO         :20.540:
```

```
REGISTRO # 00003
     NUM:EMPL       :0005:
     ANTIGDAD       :820210:
     EXTN:TEL       : 117:
     NOM:PILA       :JUAN       :
     APELLIDO       :OCHOA           :
     CALLE          :41712 VERMONT AVENUE:
     CIUDAD         :LOS ANGELES :
     ESTADO         :CA:
     COD:POSTAL     :90044:
     TEL:PART       :2135559385 :
     NO:SEG:SOC     :           :
     EXENC          : 5:
     FCHA:NAC       :0930:
     SUELDO         : 7.320:
```

```
REGISTRO # 00004
        NUM:EMPL        :0003:
        ANTIGDAD        :810118:
        EXTN:TEL        : 101:
        NOM:PILA        :THELMA     :
        APELLIDO        :ROSEKRANZ        :
        CALLE           :933 AVENUE C         :
        CIUDAD          :POMONA         :
        ESTADO          :CA:
        COD:POSTAL      :91765:
        TEL:PART        :7145556295 :
        NO:SEG:SOC      :          :
        EXENC           : 1:
        FCHA:NAC        :0319:
        SUELDO          :12.300:
```

```
REGISTRO # 00005
        NUM:EMPL        :0006:
        ANTIGDAD        :810401:
        EXTN:TEL        : 117:
        NOM:PILA        :ARNOLD     :
        APELLIDO        :PITTSINGER      :
        CALLE           :9264-A FOWLER COURT :
        CIUDAD          :DIAMOND BAR :
        ESTADO          :CA:
        COD:POSTAL      :91765:
        TEL:PART        :7145550901 :
        NO:SEG:SOC      :          :
        EXENC           : 3:
        FCHA:NAC        :0118:
        SUELDO          :14.910:
```

```
REGISTRO # 00006
     NUM:EMPL       :0007:
     ANTIGDAD       :810904:
     EXTN:TEL       : 112:
     NOM:PILA       :MARIANNE  :
     APELLIDO       :OSWALDO          :
     CALLE          :459 FRAMINGHAM PLACE:
     CIUDAD         :ONTARIO       :
     ESTADO         :CA:
     COD:POSTAL     :91761:
     TEL:PART       :7145551948 :
     NO:SEG:SOC     :          :
     EXENC          : 2:
     FCHA:NAC       :0210:
     SUELDO         : 9.370:
```

```
REGISTRO # 00007
     NUM:EMPL       :0010:
     ANTIGDAD       :811010:
     EXTN:TEL       : 109:
     NOM:PILA       :JIMMY     :
     APELLIDO       :JOHNSON          :
     CALLE          :3712 EATON PLACE     :
     CIUDAD         :AZUSA         :
     ESTADO         :CA:
     COD:POSTAL     :91702:
     TEL:PART       :2135551216 :
     NO:SEG:SOC     :          :
     EXENC          : 7:
     FCHA:NAC       :0421:
     SUELDO         : 8.355:
```

```
REGISTRO # 00008
     NUM:EMPL        :0012:
     ANTIGDAD        :811204:
     EXTN:TEL        : 105:
     NOM:PILA        :RENATE     :
     APELLIDO        :FUSBANCK        :
     CALLE           :632 STOOLER STREET   :
     CIUDAD          :LA VERNE     :
     ESTADO          :CA:
     COD:POSTAL      :91750:
     TEL:PART        :7145558003 :
     NO:SEG:SOC      :           :
     EXENC           : 3:
     FCHA:NAC        :0704:
     SUELDO          :14.160:
```

```
REGISTRO # 00009
     NUM:EMPL        :0016:
     ANTIGDAD        :821029:
     EXTN:TEL        : 106:
     NOM:PILA        :ANNAMARIE :
     APELLIDO        :OETTINGER       :
     CALLE           :208 DEUTSCH PLACE     :
     CIUDAD          :LA PUENTE    :
     ESTADO          :CA:
     COD:POSTAL      :91745:
     TEL:PART        :2135553960 :
     NO:SEG:SOC      :           :
     EXENC           : 1:
     FCHA:NAC        :1017:
     SUELDO          : 7.040:
```

```
REGISTRO # 00010
     NUM:EMPL        :0019:
     ANTIGDAD        :820228:
     EXTN:TEL        : 113:
     NOM:PILA        :ARNE          :
     APELLIDO        :SVENSON              :
     CALLE           :844 MALMO AVENUE         :
     CIUDAD          :GLENDORA        :
     ESTADO          :CA:
     COD:POSTAL      :91740:
     TEL:PART        :2135557140 :
     NO:SEG:SOC      :           :
     EXENC           : 4:
     FCHA:NAC        :0530:
     SUELDO          :12.253:
```

```
REGISTRO # 00011
     NUM:EMPL        :0024:
     ANTIGDAD        :820530:
     EXTN:TEL        : 114:
     NOM:PILA        :HANNAH        :
     APELLIDO        :LORE                 :
     CALLE           :18820 NEW MEXICO AVE:
     CIUDAD          :ALHAMBRA        :
     ESTADO          :CA:
     COD:POSTAL      :91823:
     TEL:PART        :2135558421 :
     NO:SEG:SOC      :           :
     EXENC           : 3:
     FCHA:NAC        :0728:
     SUELDO          : 9.890:
```

REGISTRO # 00012

NUM:EMPL	:0025:
ANTIGDAD	:820930:
EXTN:TEL	: 115:
NOM:PILA	:CEASAR :
APELLIDO	:FRANCK :
CALLE	:8220 AUSTRIA CIRCLE :
CIUDAD	:MONTERREY PK:
ESTADO	:CA:
COD:POSTAL	:91754:
TEL:PART	:2135554097 :
NO:SEG:SOC	: :
EXENC	: 3:
FCHA:NAC	:1210:
SUELDO	: 5.490:

REGISTRO # 00013

NUM:EMPL	:0026:
ANTIGDAD	:821127:
EXTN:TEL	: 114:
NOM:PILA	:LOU :
APELLIDO	:BAYTOVEN :
CALLE	:770 BONN STREET :
CIUDAD	:CLAREMONT :
ESTADO	:CA:
COD:POSTAL	:91711:
TEL:PART	:7145551827 :
NO:SEG:SOC	: :
EXENC	: 1:
FCHA:NAC	:1216:
SUELDO	:18.950:

```
REGISTRO # 00014
      NUM:EMPL        :0029:
      ANTIGDAD        :830119:
      EXTN:TEL        : 107:
      NOM:PILA        :JOE         :
      APELLIDO        :HIDEN            :
      CALLE           :732 ROHRAU STREET     :
      CIUDAD          :LOS ANGELES :
      ESTADO          :CA:
      COD:POSTAL      :90007:
      TEL:PART        :2135551809 :
      NO:SEG:SOC      :           :
      EXENC           : 1:
      FCHA:NAC        :0331:
      SUELDO          :18.850:
```

```
REGISTRO # 00015
      NUM:EMPL        :0030:
      ANTIGDAD        :830414:
      EXTN:TEL        : 105:
      NOM:PILA        :JOHN        :
      APELLIDO        :BOCK             :
      CALLE           :685 EISENACH PLACE    :
      CIUDAD          :WALNUT      :
      ESTADO          :CA:
      COD:POSTAL      :91789:
      TEL:PART        :7145551750 :
      NO:SEG:SOC      :           :
      EXENC           :23:
      FCHA:NAC        :0321:
      SUELDO          :18.920:
```

```
REGISTRO # 00016
      NUM:EMPL        :0029:
      ANTIGDAD        :830830:
      EXTN:TEL        : 103:
      NOM:PILA        :GEORGE     :
      APELLIDO        :HANDLE          :
      CALLE           :685 HALLE STREET      :
      CIUDAD          :IRWINDALE    :
      ESTADO          :CA:
      COD:POSTAL      :91706:
      TEL:PART        :2135551759 :
      NO:SEG:SOC      :           :
      EXENC           : 1:
      FCHA:NAC        :0223:
      SUELDO          :18.850:
```

```
REGISTRO # 00017
      NUM:EMPL        :0031:
      ANTIGDAD        :831004:
      EXTN:TEL        : 108:
      NOM:PILA        :JOHN       :
      APELLIDO        :BRAMZ           :
      CALLE           :833 HAMBURG WAY        :
      CIUDAD          :BALDWIN PARK:
      ESTADO          :CA:
      COD:POSTAL      :91706:
      TEL:PART        :2135551897 :
      NO:SEG:SOC      :           :
      EXENC           : 1:
      FCHA:NAC        :0507:
      SUELDO          :18.820:
```

```
REGISTRO # 00018
        NUM:EMPL       :0033:
        ANTIGDAD       :831120:
        EXTN:TEL       : 110:
        NOM:PILA       :WILLIE      :
        APELLIDO       :MOZART               :
        CALLE          :756 SALZBURG AVENUE :
        CIUDAD         :MONROVIA       :
        ESTADO         :CA:
        COD:POSTAL     :91016:
        TEL:PART       :2135551791 :
        NO:SEG:SOC     :           :
        EXENC          : 2:
        FCHA:NAC       :0127:
        SUELDO         :18.800:
```

ÍNDICE

EN ESTA MISMA

COLECCION

BASIC
Richard G. Peddicord

El Basic es el lenguaje de programación más importante que existe. Resulta fácil de aprender y es de aplicación universal. Quien desee utilizar un ordenador y escribir sus propios programas deberá aprender a «hablar» este lenguaje. Mediante este libro se puede dominar el Basic a través de un método preciso, bien estructurado y de sencilla comprensión. El principiante se convertirá paso a paso en un buen conocedor del Basic siguiendo los ejemplos de programación que aquí se ofrecen.

- Variables en cadena y variables numéricas
- La instrucción IF...THEN...
- PRINT USING
- Los bucles FOR...NEXT
- Tabla de conceptos y ejemplos prácticos

COMMODORE 64 BASIC

Quien posea este ordenador personal —de múltiples aplicaciones— debe llegar a dominarlo perfectamente. Este libro abrirá al principiante el camino para una óptima utilización de las aplicaciones del **Commodore 64.**

Los numerosos ejercicios que figuran en esta obra constituyen una gran ayuda para un rápido y efectivo aprendizaje.

- Sistema y aplicaciones
- Programación en Basic
- Operaciones simples, ejercicios prácticos
- Aparatos periféricos

SUPERCALC

Los programas de hojas electrónicas constituyen un instrumento imprescindible para la planificación financiera y para los cálculos, de tal forma que amplían enormemente las aplicaciones posibles de los ordenadores. Entre los programas más difundidos de este estilo se encuentra sin duda el SUPERCALC.

Este libro inicia al principiante en los conocimientos básicos del SUPERCALC y consigue que éste se familiarice con el manejo de este programa.

- Aplicaciones elementales
- Entrada de datos
- Iniciación a los comandos
- Aplicaciones avanzadas

COMMODORE 64
Michael Boom

El Commodore 64 pertenece a la nueva generación de ordenadores personales que ofrecen un elevado rendimiento. Ocupa una posición líder en el mercado español. El principiante podrá iniciarse en profundidad en las aplicaciones y peculiaridades del Commodore 64, familiarizándose con el funcionamiento del mismo.

- Sistema de manejo
- Teclado y pantalla
- Registro, memorización y recuperación de datos
- Aparatos periféricos y aplicaciones

ORDENADORES PERSONALES
Carlton Shrum

En este libro se exploran las partes integrantes de un ordenador personal. Se comentan las distintas posibles aplicaciones de los ordenadores personales en el mundo en que nos movemos. Facilita también la decisión que debe tomar el comprador frente a la fama de ordenadores que ofrece el mercado informático. Se trata de un libro de fácil comprensión y de gran utilidad para una buena decisión.

- Iniciación a los ordenadores personales.

IBM PC
Steven Manus

El ordenador personal PC de IBM es uno de los microprocesadores más populares. El IBM PC es reconocido no sólo por sus aplicaciones comerciales, sino también por sus otras múltiples aplicaciones. Este libro introduce al principiante en el terreno del IBM PC y del IBM PC XT, al tiempo que informa de modo preciso acerca del funcionamiento de ambos sistemas.

- Sistema y manejo
- Teclado y pantalla
- Registro, memorización y recuperación de datos
- Aparatos periféricos y aplicaciones

WORDSTAR

WORDSTAR está reconocido mundialmente como el mejor sistema para el proceso de textos en microordenadores. En adaptación a los modelos más diversos existen centenares de variantes.

Este libro está concebido de tal forma que cualquiera que desee familiarizarse con el «Wordstar» puede hacerlo con suma facilidad. Partiendo de los conocimientos elementales, se explica e ilustra a través de ejemplos el funcionamiento de los diferentes menús.

- Funciones elementales
- Menús iniciales y principales
- Menús secundarios
- Comandos DOT
- Mail Merge

Impreso en
T. G. Ramón Sopena, S. A.
Provenza, 95
08029 Barcelona

dBASE II

Ningún usuario de ordenadores personales podrá prescindir, a la larga, de las ventajas de los modernos sistemas de proceso de datos. dBASE II es uno de los programas más difundidos y elaborados de esta índole.

Este libro permite al principiante introducirse en el manejo del dBASE II y le facilita una visión general del gran rendimiento de este programa.

- Bases del proceso de datos
- Creación de un sistema de proceso de datos con dBASE II
- Aplicaciones y manejo
- Clasificación y recuperación



P & J

Joseph Reymann

dBASE II